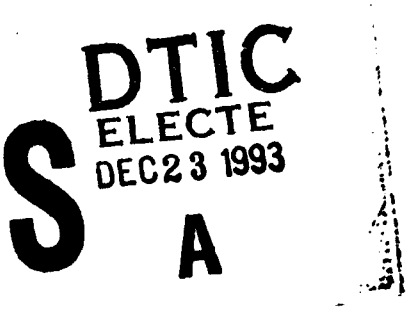

AFIT/EN-TR-93-09

Air Force Institute of Technology

Source Code for the OO1 Benchmark and the AFIT Simulation Benchmark

Timothy J. Halloran
Capt, USAF

5 November 1993

Source Code for the OO1 Benchmark and the AFIT Simulation Benchmark

Timothy J. Halloran[‡]

Abstract

This paper contains the source code for implementations of the OO1 benchmark and the AFIT simulation benchmark. OO1 benchmark implementations are listed for the Itasca, Matisse, and ObjectStore object-oriented database management systems (DBMS). An AFIT simulation benchmark implementation is listed for the ObjectStore object-oriented DBMS.

1 Introduction

As part of research into the performance of current commercial object-oriented DBMSs, we developed implementations of the OO1 benchmark and the simulation benchmark [2]. The OO1 benchmark is specified in [1]. The OO1 benchmark is a generic measure of object-oriented DBMS performance. The simulation benchmark is specified in [2]. The simulation benchmark examined the performance of object-oriented DBMSs in the simulation domain. This paper contains the source code for three implementations of the OO1 benchmark and one implementation of the simulation benchmark.

2 OO1 Benchmark

This section lists the source code for our implementations of the OO1 benchmark. Implementations were created for the Itasca, Matisse, and ObjectStore object-oriented DBMSs. Although several versions of these implementations were developed for each object-oriented DBMS, only our final implementation is shown.

All three OO1 benchmark implementations used the benchmark support software described in [2]. This common code is not listed for each implementation

2.1 Itasca Implementation

This section lists the source code for the Itasca implementation of the OO1 benchmark created for this research. The implementation was created on version 2.2 of Itasca using the Itasca C++ API. The Itasca OO1 benchmark source code builds one executable program, the *bench* program. The source files in this implementation are shown in Figure 1. The files *debug.hh*, *dice.hh*, *dice.cc*, *pmmlcg.h*, *pmmlcg.c*, *stopwtch.hh*, and *stopwtch.cc* are not listed, because they were described (with a full source code listing) in [2]. The following subsections list the source code for the other files shown in Figure 1.

[‡]The author is with the Department of Electrical and Computer Engineering (AFIT/ENG), Air Force Institute of Technology, 2950 P ST, Wright-Patterson AFB, OH 45433-7765.

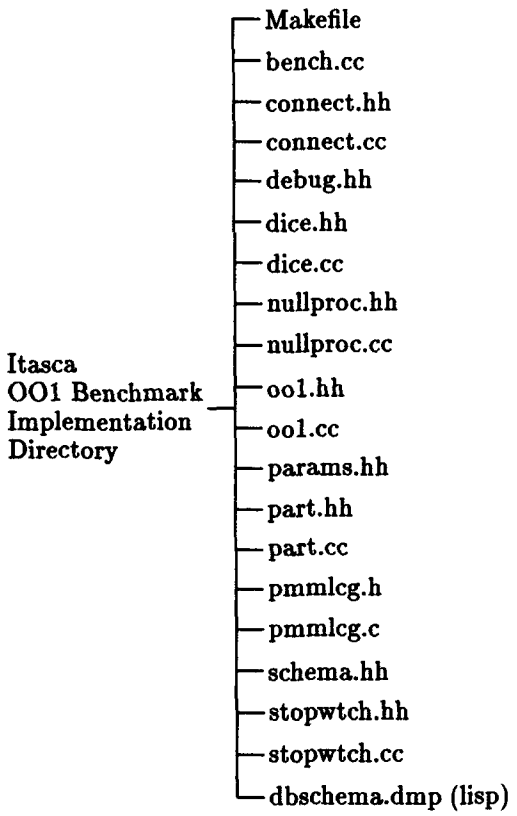

Figure 1: Itasca OO1 Benchmark Source Code Files

2.1.1 Makefile

The *Makefile* for the Itasca OO1 benchmark source code is listed below.

```
CCC=CC
#########################################################################
#
# If debug output is desired uncomment the "CCFLAGS" definition with
#  "-DDEBUG" in it.  Only one definition should be uncommented.
#
#CCFLAGS=-DDEBUG -g
CCFLAGS=
#
#########################################################################
LDFLAGS =-Bstatic
INCLUDES=-I. -I/usr/itasca/include
LIBS=-L/usr/itasca/lib -lItascacpp

all: bench

.c.o:
        $(CCC) $(CCFLAGS) $(INCLUDES) -c $<

.cc.o:
        $(CCC) $(CCFLAGS) $(INCLUDES) -c $<

pmmlcg.o: pmmlcg.c pmmlcg.h

dice.o: dice.cc debug.hh dice.hh pmmlcg.h

stopwtch.o: stopwtch.cc debug.hh stopwtch.hh

nullproc.o: nullproc.cc dice.hh

part.o: part.cc debug.hh nullproc.hh params.hh schema.hh

connect.o: connect.cc debug.hh schema.hh

oo1.o: oo1.cc debug.hh dice.hh nullproc.hh oo1.hh params.hh
  schema.hh stopwtch.hh

bench.o: bench.cc oo1.hh schema.hh debug.hh

bench: bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o
        $(CCC) $(LDFLAGS) -o bench \
         bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o \
         $(LIBS)

clean:
        rm -f *.o bench
```


2.1.2 bench.cc

The source code for the file *bench.cc* is listed below. This file provides a driver for the benchmark. It consists of a main program which connects to the Itasca database, then performs the benchmark operation specified on the command line.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
```

```
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 bench.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993

 This program performs all the benchmark measurements for the 001 benchmark.
  The following benchmark operations are supported: database creation (and
  load), lookup, forward traversal, reverse traversal, insert, and database
  clear (and destroy).

 Revisions:
 20 Aug 1993 -TJH- Changed out random number generator.
*/
#include<stdio.h>
#include<stdlib.h>
#include<debug.hh>
#include"schema.hh"
#include"oo1.hh"

#define TRUE 1
#define FALSE 0
////////////////////////////////////////////////////////////////////////
main( int argc, char **argv)
{
  DEBUG_INIT( "MAIN" , "main" );
  DEBUG_OUT( "entering", 1 );

  // check the command line arguments
  if ( argc < 7 ) {
    fprintf( stderr, "bench [operation] [db server] [username] "
      "[password] [# of parts] [random stream]\n" );
    return 1;  // exit the program
  }
  char *operation = argv[1];
  if ( !( !strcmp( operation, "load" ) || !strcmp( operation, "lookup" ) ||
    !strcmp( operation, "ftrav" ) || !strcmp( operation, "rtrav" ) ||
    !strcmp( operation, "insert" ) || !strcmp( operation, "clear" ) ) ) {
    fprintf( stderr, "ERROR[main] the operation must be \"load\", \"lookup\", "
      "\"ftrav\", \"rtrav\", \"insert\", or \"clear\"\n" );
    return 1;  // exit the program
  }
  char *db_server = argv[2];
  char *username = argv[3];
  char *password = argv[4];
  int num_parts = atoi( argv[5] );
  if ( ( num_parts != 20000 ) && ( num_parts != 200000 ) )
    fprintf( stderr, "WARNING[main] the number of parts is not 20,000 or 200,000\n" );
  int stream = atoi( argv[6] );
  if ( ( stream < 1 ) || ( stream > 100 ) ) {
    fprintf( stderr, "ERROR[main] the stream is not between 1 and 100\n" );
    return 1;  // exit the program
  }

  // output a program title
  printf( "001 BENCHMARK [ITASCA] Air Force Institute of Technology\n" );
```

```
  printf( " Itasca database user \"%s\" on %s with %d parts (random stream %d).\n",
    username, db_server, num_parts, stream );

  // open a connection to the Itasca database
  if ( ITASCA::connect( db_server, "itasca-interface", username, password ) ) {
    fprintf( stderr, "ERROR[main] unable to connect to database \"%s:%s\" on %s\n",
      username, password, db_server );
    return 1;  // exit the program
  }

  // create a single 001 benchmark object
  oo1 oo1_benchmark( num_parts, stream );

  // perform the benchmark measurement requested on the command line
  if ( !strcmp( operation, "load" ) ) {
    printf( " LOAD\n" );
    oo1_benchmark.load( );
  }
  else if ( !strcmp( operation, "lookup" ) ) {
    printf( " LOOKUP\n" );
    oo1_benchmark.lookup_measure( );
  }
  else if ( !strcmp( operation, "ftrav" ) ) {
    printf( " FORWARD TRAVERSAL\n" );
    oo1_benchmark.forward_traversal_measure( );
  }
  else if ( !strcmp( operation, "rtrav" ) ) {
    printf( " REVERSE TRAVERSAL\n" );
    oo1_benchmark.reverse_traversal_measure( );
  }
  else if ( !strcmp( operation, "insert" ) ) {
    printf( " INSERT\n" );
    oo1_benchmark.insert_measure( );
  }
  else if ( !strcmp( operation, "clear" ) ) {
    printf( " CLEAR\n" );
    oo1_benchmark.clear( );
  }
  ITASCA::commit( );
  ITASCA::disconnect( );
}
////////////////////////////////////////////////////////////////////////
```

2.1.3 connect.hh

The source code for the file *connect.hh* is listed below. This file defines function prototypes for the C++ class methods of the *Oo1Connection* class. The *Oo1Connection* class is fully defined in the file *schema.hh*. The *connect.hh* file is included into the file *schema.hh* during the definition of the *Oo1Connection* class. Itasca Customer Support recommended including definitions of C++ class methods, rather than just adding them directly to the *schema.hh* file (as is recommended in the Itasca documentation [3]) to simplify maintenance.

```
#ifndef __CONNECT_HH
#define __CONNECT_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####
```

```
"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993

 This file only contains the function prototypes for the connect class
  methods.  This file is included by the "schema.hh" file (which was
  automatically generated by Itasca).  Use of this file ensures that the
  changes necessary to the "schema.hh" file are not lost each time it is
  regenerated.
*/
void set_Oo1Connection( Oo1Part *new_from, Oo1Part *new_to,
                        char *new_type, int new_length );
void clear_Oo1Connection( );
#endif __CONNECT_HH
```

2.1.4 connect.cc

The source code for the file *connect.cc* is listed below. This file implements the non-persistent methods for the *Oo1Connection* class. The *Oo1Connection* class is fully defined in the file *schema.hh*.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993
*/
#include<debug.hh>
#include"schema.hh"
////////////////////////////////////////////////////////////////////////
void Oo1Connection::set_Oo1Connection( Oo1Part *new_from, Oo1Part *new_to,
  char *new_type, int new_length )
{
  DEBUG_INIT( "CONNECTION" , "Oo1Connection::set_Oo1Connection" );
  DEBUG_OUT( "entering", 1 );

  from = new_from;
  from->connectedTo.addToSet( this );  // also set the inverse
  to = new_to;
  to->connectedFrom.addToSet( this );  // also set the inverse
  type = new_type;
  cLength = new_length;
}
////////////////////////////////////////////////////////////////////////'
```

```
void Oo1Connection::clear_Oo1Connection( )
{
  DEBUG_INIT( "CONNECTION" , "Oo1Connection::clear_Oo1Connection" );
  DEBUG_OUT( "entering", 1 );

  from->connectedTo.removeFromSet( this );
  to->connectedFrom.removeFromSet( this );
}
//////////////////////////////////////////////////////////////////////////
```

2.1.5 nullproc.hh

The source code for the file *nullproc.hh* is listed below. This file defines the null procedures which are required to be called at certain points in the OO1 benchmark.

```
#ifndef __NULLPROC_HH
#define __NULLPROC_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                   "null_procedure".
*/
void null_procedure( int x, int y, char *type );
void null_procedure_get_x_y( int& new_x, int& new_y, int stream );
#endif __NULLPROC_HH
```

2.1.6 nullproc.cc

The source code for the file *nullproc.cc* is listed below. This file implements the null procedures which are required to be called at certain points in the OO1 benchmark.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
```

```
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                  "null_procedure".
*/
#include<stdio.h>
#include<dice.hh>

#define TRUE  1
#define FALSE 0
static int debug = FALSE;
////////////////////////////////////////////////////////////////////////
void null_procedure( int x, int y, char *type )
{
  // this procedure does nothing (a null procedure)
  // (a special debug scheme is used here so that even a very smart
  //  compiler will still make this procedure call)
  if (debug)
    printf( "[null_procedure] %5d %5d %s\n", x, y, type );
}
////////////////////////////////////////////////////////////////////////
void null_procedure_get_x_y( int& new_x, int& new_y, int stream )
{
  new_x = (int)roll( 0, 99999, stream );
  new_y = (int)roll( 0, 99999, stream );
}
////////////////////////////////////////////////////////////////////////
```

2.1.7 oo1.hh

The source code for the file *oo1.hh* is listed below. This file defines the *OO1* class. This class implements the benchmark measurements and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
#ifndef __OO1_HH
#define __OO1_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993
*/
```

```
#include"params.hh"

struct benchmark_results_type {
  double elapsed_time;
  double normalized_elapsed_time;
  int num_parts_connected;
} ;

class oo1 {
  int num_nodes_visited_in_forward_traversal;
  int num_parts;
  benchmark_results_type results[NUM_BENCHMARK_ITERATIONS];
  int stream;  // for the random number generator
  Oo1Part *create_a_part( int part_id );
  void create_connections( Oo1Part *db_part, int num_parts );
  void report_results( int measurement );
public:
  oo1( int num_parts, int stream );
  void clear( );
  void forward_traversal_measure( );
  void insert_measure( );
  void load( );
  void lookup_measure( );
  void reverse_traversal_measure( );
};
#endif __OO1_HH
```

2.1.8 oo1.cc

The source code for the file *oo1.cc* is listed below. This file implements the *OO1* class. This class implements the benchmark measurements and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993

 Revisions:
  19 Aug 1993 -TJH- Changed the "report_results( )" function to provide more
                    concise output.
  20 Aug 1993 -TJH- Changed out random number generator.
  21 Aug 1993 -TJH- Changed the "load( )" function to perform intermediate
                    commits (rather than trying to load the entire database
                    in a single transaction). Loading the entire database
                    was causing this program to use to much system memory.
  07 Sep 1993 -TJH- Changed the code to fix several memory leaks.  ITASCA
                    does not manage the transient memory associated with
```

```
                    persistant objects as well as I thought, the application
                    is responsible for freeing it in most cases.
*/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
extern "C" int strftime(...);  // not defined in the <time.h> header file
#include<dice.hh>
#include<stopwtch.hh>
#include<debug.hh>
#include"schema.hh"
#include"oo1.hh"
#include"nullproc.hh"
#include"params.hh"

enum { INSERT, FORWARD_TRAVERSAL, LOOKUP, REVERSE_TRAVERSAL };
static char *part_types[10] = {
  "part-type0", "part-type1", "part-type2", "part-type3", "part-type4",
  "part-type5", "part-type6", "part-type7", "part-type8", "part-type9"
};
////////////////////////////////////////////////////////////////////////
oo1::oo1( int num_parts, int stream )
{
  DEBUG_INIT( "OO1" , "oo1::oo1" );
  DEBUG_OUT( "entering", 1 );

  // save the number of parts in the database
  this->num_parts = num_parts;

  // save the stream number for use with the random number generator
  this->stream = stream;

  // calculate the number of nodes which will be visited in a forward traversal
  // (this value will be used to normalize the measurements
  //  from the reverse traversal)
  int nodes_at_current_level = NUM_CONNECTIONS;
  num_nodes_visited_in_forward_traversal = 1 + NUM_CONNECTIONS;
  for ( int i = 2 ; i <= TRAVERSAL_DEPTH ; i++ ) {
    nodes_at_current_level *= NUM_CONNECTIONS;
    num_nodes_visited_in_forward_traversal += nodes_at_current_level;
  }
  DEBUG_OUT( sprintf( BUG, "number of nodes which will be visited in the "
    "forward traversal %d", num_nodes_visited_in_forward_traversal ), 2 );
}
////////////////////////////////////////////////////////////////////////
void oo1::clear( )
{
  DEBUG_INIT( "OO1" , "oo1::clear" );
  DEBUG_OUT( "entering", 1 );

  OoiPartClassObject part_class;
  OoiConnectionClassObject connection_class;

  part_class.pDelete( "t" );  // delete all the parts
  connection_class.pDelete( "t" );  // delete all the connections

  ITASCA::commit( );

  // note that this procedure doesn't delete the schema
  DEBUG_OUT( "all parts and connections deleted (but the schema remains)", 2 );
}
////////////////////////////////////////////////////////////////////////
OoiPart *oo1::create_a_part( int part_id )  // private
{
```

```
  DEBUG_INIT( "001" , "oo1::create_a_part" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *new_part;
  int new_x;
  int new_y;
  char text_build_date[10];

  // create a new part in the database
  char *new_type = part_types[roll( 0, 9, stream )];
  null_procedure_get_x_y( new_x, new_y, stream );  // null procedure
  long new_build_datetime = roll( JAN_1_1980, JAN_1_1990, stream );
  struct tm *new_tm_timedate = localtime( &new_build_datetime );
  strftime( text_build_date, 10, "%d%b%Y", new_tm_timedate );
  DEBUG_OUT( sprintf( BUG, "adding part %d (%s %5d %5d %s)",
    part_id, new_type, new_x, new_y, text_build_date ), 2 );
  new_part = new Oo1Part;
  if ( !new_part->Okay( ) ) {
    fprintf( stderr, "ERROR[oo1::create_a_part] Itasca failed to create "
      "part number %d\n", part_id );
    ITASCA::abort( );
    ITASCA::disconnect( );
    exit( 1 );
  }
  new_part->set_Oo1Part( part_id, new_type, new_x, new_y, new_build_datetime );
  return( new_part );
}
//////////////////////////////////////////////////////////////////////
void oo1::create_connections( Oo1Part *db_part, int num_parts )  // private
{
  DEBUG_INIT( "001" , "oo1::create_connections" );
  DEBUG_OUT( "entering", 1 );
  int cpart_id;
  Oo1Part *db_cpart;
  Oo1Connection *new_connection;
  int new_length;
  char *new_type;
  char query_expression[80];
  int status;

  // create NUM_CONNECTIONS from the part "db_part"
  for ( int c = 1 ; c <= NUM_CONNECTIONS ; c++ ) {
    if ( roll( 1, 10, stream ) > LOCALITY_OF_REFERENCE ) {
      // 90% of the time create connection to the closest 1% of parts
      //  (this is true when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = db_part->id + (int)roll( 1, (long)( num_parts / 100 ), stream )
        - 1 - (int)( num_parts / 200 );
      // "double up" at the ends (to stay in the part id range)
      if ( cpart_id < (int)( num_parts / 200 ) )
        cpart_id = cpart_id + (int)( num_parts / 200 );
      if ( cpart_id > ( num_parts - (int)( num_parts / 200 ) ) )
        cpart_id = cpart_id - (int)( num_parts / 200 );
    }
    else {
      // 10% of the time create connection to any part 1..num_parts
      //  (this is true when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = (int)roll( 1, (long)num_parts, stream );
    }

    // create the connection in the database
    // lookup the part we are connecting to with an Itasca query
    db_cpart = NULL;
    sprintf( query_expression, "(equal id %d)", cpart_id );
    status = Oo1Part::selectAny( &db_cpart,
      QUERY_EXPRESSION, query_expression,
```

```
      END_ARGS );
    if ( ( status != ITASCA_OK ) || ( db_cpart == NULL ) ) {
      fprintf( stderr, "ERROR[oo1::create_connections] Itasca failed query\n" );
      ITASCA::abort( );
      ITASCA::disconnect( );
      exit( 1 );
    }
    new_type = part_types[roll( 0, 9, stream )];
    new_length = (int)roll( 0, 99999, stream );
    DEBUG_OUT( sprintf( BUG, "connecting part %d to part %d (%s %5d)",
      (int)db_part->id, (int)db_cpart->id, new_type, new_length ), 2 );
    new_connection = new Oo1Connection;
    if ( !new_connection->Okay( ) ) {
      fprintf( stderr, "ERROR[oo1::create_connections] Itasca failed to "
        "create a connection\n" );
      ITASCA::abort( );
      ITASCA::disconnect( );
      exit( 1 );
    }
    new_connection->set_Oo1Connection( db_part, db_cpart, new_type, new_length );

    // the application must remove transient memory associated with
    //  persistance objects
    delete new_connection;
    delete db_cpart;
  }
}
//////////////////////////////////////////////////////////////////////
void oo1::forward_traversal_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::forward_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *db_part;
  int part_id;
  char query_expression[80];
  int status;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch traversal_timer;
    ITASCA::commit( );  // start a new transaction
    traversal_timer.start( );

    for ( int j = 1 ; j <= NUM_FORWARD_TRAVERSAL ; j++ ) {

      // lookup a random part
      part_id = (int)roll( 1, (long)num_parts, stream );
      // lookup the part with an Itasca query
      sprintf ( query_expression, "(equal id %d)", part_id );
      status = Oo1Part::selectAny( &db_part,
        QUERY_EXPRESSION, query_expression,
        END_ARGS );
      if ( ( status != ITASCA_OK ) || ( db_part == NULL ) ) {
        fprintf( stderr, "ERROR[oo1::forward_traversal_measure] Itasca failed "
          "query\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }

      // find all the parts connected to this part (up to TRAVERSAL_DEPTH)
      results[i - 1].num_parts_connected = db_part->forward_traversal( 0 );
```

```
      DEBUG_OUT( sprintf( BUG, "found %d parts connected to part %d",
        results[i - 1].num_parts_connected, (int)db_part->id ), 2 );

      // the application must remove transient memory associated with
      //  persistance objects
      delete db_part;
    }

    // stop the timer and save the elapsed time
    ITASCA::commit( );
    results[i - 1].elapsed_time = traversal_timer.stop( );
  }

  // report the benchmark results
  report_results( FORWARD_TRAVERSAL );
}
////////////////////////////////////////////////////////////////////////
void oo1::insert_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::insert_measure" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *db_part;
  Iboolean deleted;
  ItascaSet part_set;
  char query_expression[80];
  int status;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch insert_timer;
    ITASCA::commit( );  // start a new transaction
    insert_timer.start( );

    // insert parts into the database and call a "null" procedure
    //  to get the x and y position for each insert
    // (the "null" procedure is called by the "create_a_part( )" function)
    for ( int p = num_parts + 1 ; p <= num_parts + NUM_INSERT ; p++ )
      create_connections( create_a_part( p ), p );

    // stop the timer and save the elapsed time
    ITASCA::commit( );
    results[i - 1].elapsed_time = insert_timer.stop( );

    // remove the inserted parts and connections
    int largest_original_part_id = num_parts;
    Oo1PartClassObject part_class;
    // determine the extra parts with an Itasca query
    sprintf( query_expression, "( > id %d)", largest_original_part_id );
    status = Oo1Part::select( part_set,
      QUERY_EXPRESSION, query_expression,
      END_ARGS );
    if ( ( status != ITASCA_OK ) || ( part_set.size( ) == 0 ) ){
      fprintf( stderr, "ERROR[oo1::insert_measure] Itasca failed query\n" );
      ITASCA::abort( );
      ITASCA::disconnect( );
      exit( 1 );
    }
    DEBUG_OUT( sprintf( BUG, "now %d extra parts in the database",
      part_set.size( ) ), 2 );
    // delete all the extra parts (and corresponding connections)
    Oo1PartSetIterator iterator( part_set );
    while ( db_part = iterator( ) ) {
```

```
      db_part->clear_Oo1Part( );  // clear the part for delete
      db_part->deleteObject( &deleted );
      if ( !deleted ) {
        fprintf( stderr, "ERROR[oo1::insert_measure] Itasca failed delete\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }
    }
    ITASCA::commit( );
  }

  // report the benchmark results
  report_results( INSERT );
}
////////////////////////////////////////////////////////////////////////
void oo1::load( )
{
  DEBUG_INIT( "001" , "oo1::load" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *db_part;
  ItascaSet part_set;
  int status;

  // time the creation of the 001 database
  stopwatch db_creation_timer;
  ITASCA::commit( );  // start a new transaction
  db_creation_timer.start( );

  // create "num_part" parts
  DEBUG_OUT( "creating parts", 2 );
  for ( int p = 1 ; p <= num_parts ; p++ ) {
    delete create_a_part( p );
    if ( !( p % NUM_COMMIT ) ) {
      ITASCA::commit( );
      DEBUG_OUT( "performing commit", 2 );
    }
  }

  // create an index to optimize lookup by part id
  Oo1PartClassObject part_class;
  part_class.makeIndex( "id", END_ARGS );

  // create connections for each part
  status = Oo1Part::select( part_set, END_ARGS );
  if ( ( status != ITASCA_OK ) || ( part_set.size( ) == 0 ) ){
    fprintf( stderr, "ERROR[oo1::load] Itasca failed query\n" );
    ITASCA::abort( );
    ITASCA::disconnect( );
    exit( 1 );
  }
  p = 1;  // reset the counter for commits
  Oo1PartSetIterator iterator( part_set );
  while ( db_part = iterator( ) ) {
    create_connections( db_part, num_parts );
    if ( !( p++ % NUM_COMMIT ) ) {
      ITASCA::commit( );
      DEBUG_OUT( "performing commit", 2 );
    }
  }

  // report the time it took to load the database
  ITASCA::commit( );
  double total_elapsed_time = db_creation_timer.stop( );
```

```
  printf( " %.3f sec\n", total_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::lookup_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::lookup_measure" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *db_part;
  int part_id;
  char query_expression[80];
  int status;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch lookup_timer;
    ITASCA::commit( );  // start a new transaction
    lookup_timer.start( );

    // lookup parts in the database and call a null procedure for each lookup
    for ( int j = 1 ; j <= NUM_LOOKUP ; j++ ) {

      // lookup a random part
      part_id = (int)roll( 1, (long)num_parts, stream );
      // lookup the part with an Itasca query
      sprintf ( query_expression, "(equal id %d)", part_id );
      status = Oo1Part::selectAny( &db_part,
        QUERY_EXPRESSION, query_expression,
        END_ARGS );
      if ( ( status != ITASCA_OK ) || ( db_part == NULL ) ) {
        fprintf( stderr, "ERROR[oo1::lookup_measure] Itasca failed query\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }
      DEBUG_OUT( sprintf( BUG, "looked up part %d", (int)db_part->id ), 2 );

      // call the required null procedure
      null_procedure( db_part->x, db_part->y, db_part->type );

      // the application must remove transient memory associated with
      //  persistance objects
      delete db_part;
    }

    // stop the timer and save the elapsed time
    ITASCA::commit( );
    results[i - 1].elapsed_time = lookup_timer.stop( );
  }

  // report the benchmark results
  report_results( LOOKUP );
}
////////////////////////////////////////////////////////////////////////
void oo1::report_results( int measurement )  // private
{
  { // two debug INITs used in this function so hide this one from the other
    DEBUG_INIT( "OO1" , "oo1::report_results" );
    DEBUG_OUT( "entering", 1 );
  }
  double cold_elapsed_time;
  double total_after_first_iteration = 0.0;
  double warm_elapsed_time;
```

```
  // report the benchmark results
  DEBUG_INIT( "RESULTS" , "oo1::report_results" );

  // determine the cold time
  if ( measurement == REVERSE_TRAVERSAL )
    // the reverse traversal times are normalized
    cold_elapsed_time = results[0].normalized_elapsed_time;
  else
    cold_elapsed_time = results[0].elapsed_time;

  // calculate the warm time
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // output detailed results if the "RESULTS" environment variable is set
    //  to at least a level of 1 (i.e. setenv RESULTS 1)
    DEBUG_OUT( sprintf( BUG, "iteration %2d elapsed time %.3f sec", i,
      results[i - 1].elapsed_time ), 1 );
    if ( measurement == FORWARD_TRAVERSAL ) {
      DEBUG_OUT( sprintf( BUG, "(parts found %d)",
        results[i - 1].num_parts_connected ), 1 );
    }
    if ( measurement == REVERSE_TRAVERSAL ) {
      DEBUG_OUT( sprintf( BUG, "(parts found %d normalized time %.3f sec)",
        results[i - 1].num_parts_connected,
        results[i - 1].normalized_elapsed_time ), 1 );
    }

    // add up the elapsed times (for all iterations after the first)
    if (i != 1)
      if ( measurement == REVERSE_TRAVERSAL )
        total_after_first_iteration += results[i - 1].normalized_elapsed_time;
      else
        total_after_first_iteration += results[i - 1].elapsed_time;
  }
  warm_elapsed_time = total_after_first_iteration / ( NUM_BENCHMARK_ITERATIONS - 1 );

  // output a quick summary of all the times
  for ( i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {
    printf( " %7.3f", results[i - 1].elapsed_time );
  }
  // output the cold and warm results
  printf( " C %7.3f", cold_elapsed_time );
  printf( " W %7.3f\n", warm_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::reverse_traversal_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::reverse_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  Oo1Part *db_part;
  int part_id;
  char query_expression[80];
  int status;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch traversal_timer;
    ITASCA::commit( );  // start a new transaction
    traversal_timer.start( );

    for ( int j = 1 ; j <= NUM_REVERSE_TRAVERSAL ; j++ ) {
```

```
      // lookup a random part
      part_id = (int)roll( 1, (long)num_parts, stream );
      // lookup the part with an Itasca query
      sprintf ( query_expression, "(equal id %d)", part_id );
      status = OoiPart::selectAny( &db_part,
        QUERY_EXPRESSION, query_expression,
        END_ARGS );
      if ( ( status != ITASCA_OK ) || ( db_part == NULL ) ) {
        fprintf( stderr, "ERROR[oo1::reverse_traversal_measure] Itasca failed "
          " query\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }

      // find all the parts which connect to this part (up to TRAVERSAL_DEPTH)
      results[i - 1].num_parts_connected = db_part->reverse_traversal( 0 );
      DEBUG_OUT( sprintf( BUG, "found %d parts which connect to part %d",
        results[i - 1].num_parts_connected, (int)db_part->id ), 2 );

      // the application must remove transient memory associated with
      //  persistance objects
      delete db_part;
    }

    // stop the timer and save the elapsed time
    ITASCA::commit( );
    results[i - 1].elapsed_time = traversal_timer.stop( );
    // the results for the reverse traversal are normalized so that they
    //  may be compared to the forward traversal
    // (multiply the time by num_nodes_visited_in_forward_traversal/N,
    //  where N is the number of nodes actually visited in the reverse
    //  traversal)
    results[i - 1].normalized_elapsed_time = results[i - 1].elapsed_time *
      ( (double)num_nodes_visited_in_forward_traversal /
      (double)results[i - 1].num_parts_connected );
  }

  // report the benchmark results
  report_results( REVERSE_TRAVERSAL );
}
////////////////////////////////////////////////////////////////////////
```

2.1.9 params.hh

The source code for the file *params.hh* is listed below. This file defines several constant parameters which the OO1 benchmark implementation requires.

```
#ifndef __PARAMS_HH
#define __PARAMS_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. [illegible] and J. Skeen
                              Sun Microsystems
```

```
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 params.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993

 Revisions:
 21 Aug 1993 -TJH- Added Itasca specific parameter "NUM_COMMIT".  This
                   parameter specifies the number of parts (connections)
                   which will be created during the load operation before
                   a commit is done.
*/
// definition of the number of connections for each part (3 for 001)
#define NUM_CONNECTIONS 3
// definition of the depth of the forward and reverse traversals (7 for 001)
#define TRAVERSAL_DEPTH 7
// definition of the locality of reference value (1 for 001 which means that
//  90% of the connections are randomly selected among the 1% of the parts
//  which are closest (in terms of part id), and the rest of the connections
//  are made to any randomly selected part.
#define LOCALITY_OF_REFERENCE 1
// definitions to create a 10 year range of dates
#define JAN_1_1980 315550800
#define JAN_1_1990 631170000
// 001 benchmark parameters
#define NUM_BENCHMARK_ITERATIONS 10
#define NUM_LOOKUP 1000
#define NUM_FORWARD_TRAVERSAL 1
#define NUM_REVERSE_TRAVERSAL 1
#define NUM_INSERT 100
// Itasca parameters
#define NUM_COMMIT 100
#endif __PARAMS_HH
```

2.1.10 part.hh

The source code for the file *part.hh* is listed below. This file defines function prototypes for the C++ class methods of the *Oo1Part* class. The *Oo1Part* class is fully defined in the file *schema.hh*. The *part.hh* file is included into the file *schema.hh* during the definition of the *Oo1Part* class. Itasca Customer Support recommended including definitions of C++ class methods, rather than just adding them directly to the *schema.hh* file (as is recommended in the Itasca documentation [3]) to simplify maintenance.

```
#ifndef __PART_HH
#define __PART_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 part.hh

 Air Force Institute of Technology
```

```
 Timothy J. Halloran
 26 Jul 1993

 This file only contains the function prototypes for the part class
  methods.  This file is included by the "schema.hh" file (which was
  automatically generated by Itasca).  Use of this file ensures that the
  changes necessary to the "schema.hh" file are not lost each time it is
  regenerated.
*/
void set_Oo1Part( int new_id, char *new_type, int new_x, int new_y, long new_build );
void clear_Oo1Part( );
int forward_traversal( int current_level );
int reverse_traversal( int current_level );
#endif __PART_HH
```

2.1.11 part.cc

The source code for the file *part.cc* is listed below. This file implements the non-persistent methods for the *Oo1Part* class. The *Oo1Part* class is fully defined in the file *schema.hh*.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    ITASCA
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 part.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 26 Jul 1993
*/
#include<stdlib.h>
#include<debug.hh>
#include"schema.hh"
#include"nullproc.hh"
#include"params.hh"
////////////////////////////////////////////////////////////////////////
void Oo1Part::set_Oo1Part( int new_id, char *new_type, int new_x, int new_y,
  long new_build )
{
  DEBUG_INIT( "PART" , "Oo1Part::set_Oo1Part" );
  DEBUG_OUT( "entering", 1 );

  id = new_id;
  type = new_type;
  x = new_x;
  y = new_y;
  build = (int)new_build;  // int and long are both 4 bytes on sun4
}
////////////////////////////////////////////////////////////////////////
void Oo1Part::clear_Oo1Part( )
{
  DEBUG_INIT( "PART" , "Oo1Part::clear_Oo1Part" );
  DEBUG_OUT( "entering", 1 );
```

```
  OoiConnection *c;
  Iboolean deleted;

  DEBUG_OUT( sprintf( BUG, "clearing part %d", (int)id ), 2 );

  DEBUG_OUT( sprintf( BUG, "deleting the %d \"connected_to\" connections",
    connectedTo->size( ) ), 2 );
  // delete all the connections from this part to other parts
  {
    OoiConnectionSetIterator iterator( this->connectedTo );
    while ( c = iterator( ) ) {
      c->clear_OoiConnection( );  // clear the connection for delete
      c->deleteObject( &deleted );
      if ( !deleted ) {
        fprintf( stderr, "ERROR[OoiPart::clear_OoiPart] Itasca failed delete\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }
    }
  }
  DEBUG_OUT( sprintf( BUG, "deleting the %d \"connected_from\" connections",
    connectedFrom->size( ) ), 2 );
  // delete all the connections to this part from other parts
  {
    OoiConnectionSetIterator iterator( this->connectedFrom );
    while ( c = iterator( ) ) {
      c->clear_OoiConnection( );  // clear the connection for delete
      c->deleteObject( &deleted );
      if ( !deleted ) {
        fprintf( stderr, "ERROR[OoiPart::clear_OoiPart] Itasca failed delete\n" );
        ITASCA::abort( );
        ITASCA::disconnect( );
        exit( 1 );
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////
int OoiPart::forward_traversal( int current_level )
{
  DEBUG_INIT( "PART" , "OoiPart::forward_traversal" );
  DEBUG_OUT( "entering", 1 );
  OoiConnection *c;

  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)",
    (int)this->id, current_level ), 2 );
  // call the required null procedure
  null_procedure( this->x, this->y, this->type );
  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {
    OoiConnectionSetIterator iterator( this->connectedTo );
    while ( c = iterator( ) ) {
      num_parts_connected += c->to->forward_traversal( current_level + 1 );
    }
  }
  return ( num_parts_connected + 1 );
}
////////////////////////////////////////////////////////////////////////
int OoiPart::reverse_traversal( int current_level )
{
  DEBUG_INIT( "PART" , "OoiPart::reverse_traversal" );
  DEBUG_OUT( "entering", 1 );
  OoiConnection *c;
```

```
  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)",
    (int)this->id, current_level ), 2 );
  // call the required null procedure
  null_procedure( this->x, this->y, this->type );
  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {
    Oo1ConnectionSetIterator iterator( this->connectedFrom );
    while ( c = iterator( ) ) {
      num_parts_connected += c->from->reverse_traversal( current_level + 1 );
    }
  }
  return ( num_parts_connected + 1 );
}
//////////////////////////////////////////////////////////////////////////
```

2.1.12 schema.hh

The source code for the file *schema.hh* is listed below. This file contains the definitions of all the C++ classes which are persistent in the Itasca database. The file was automatically generated by the Itasca Dynamic Schema Editor. The file includes the files *connect.hh* and *part.hh* which define the non-persistent C++ methods for the *Oo1Connection* and *Oo1Part* classes. This file does not create the schema in the Itasca database, the file *dbschema.dmp* is used to to that. The *dbschema.dmp* file must be loaded into Itasca prior to any C++ program compiled with this file to execute properly.

```
//
// schema.hh
//
// Tue Aug 17 16:00:25 1993
//
// Copyright 1992 ITASCA Systems, Inc.
//
// THIS FILE WAS AUTOMATICALLY GENERATED.
// DO NOT EDIT THIS FILE.

#include "itascacpp.h"

class BenchmarkClassObject;
class Benchmark;
class Oo1AbstractConnectionClassObject;
class Oo1AbstractConnection;
class Oo1PartClassObject;
class Oo1Part;
class Oo1AbstractConnectionSetAttribute;
class Oo1ConnectionClassObject;
class Oo1Connection;
class Oo1PartAttribute;

class BenchmarkClassObject : public virtual ItascaClassObject {
public:

// Attributes and Methods

// Constructors
BenchmarkClassObject(const ItascaOID &new_oid)
: ItascaClassObject(new_oid, "BENCHMARK") {
classObjectOidCtor(new_oid, "BENCHMARK");
}
BenchmarkClassObject()
: ItascaClassObject(NULL, "BENCHMARK") {
classObjectCtor("BENCHMARK");
}
```

```
};

ItascaClassSetIter(BenchmarkClassObjectSetIterator, BenchmarkClassObject);


class Benchmark : public virtual ItascaObject {
public:
// Override functions (handled in-line via macros)
ItascaOverrideCopyObject(Benchmark)
ItascaOverrideGenericVersion(Benchmark)
ItascaOverrideMakeVersion(Benchmark)
ItascaOverrideObjectVersion(Benchmark)
ItascaOverrideParentVersion(Benchmark)
ItascaOverrideChildVersions(Benchmark)

// Select functions (handled in-line via macros)
ItascaOverrideSelectAnySimple(Benchmark,"BENCHMARK")
ItascaOverrideSelectAnyStar(Benchmark,"BENCHMARK")
ItascaOverrideSelectSimple(Benchmark,"BENCHMARK")
ItascaOverrideSelectStar(Benchmark,"BENCHMARK")

// Attributes and Methods

// Constructor which creates a new instance in ITASCA DB.
Benchmark()
: ItascaObject(NULL, "BENCHMARK") {
itascaObjectCtor("BENCHMARK");
}

// Constructor which does NOT create a new instance.
Benchmark(const ItascaOID &new_oid)
: ItascaObject(new_oid, "BENCHMARK") {
itascaObjectOidCtor(new_oid,"BENCHMARK");
}

// Copy constructor.
Benchmark(const Benchmark &src_obj)
: ItascaObject(NULL, "BENCHMARK") {
objStatus = src_obj.baseCopyObject(itasca_oid);
};
};

ItascaClassSetIter(BenchmarkSetIterator, Benchmark);


class Oo1AbstractConnectionClassObject :  public virtual BenchmarkClassObject {
public:

// Attributes and Methods

// Constructors
Oo1AbstractConnectionClassObject(const ItascaOID &new_oid)
: BenchmarkClassObject(new_oid),
ItascaClassObject(new_oid, "OO1-ABSTRACT-CONNECTION") {
classObjectOidCtor(new_oid, "OO1-ABSTRACT-CONNECTION");
}
Oo1AbstractConnectionClassObject()
: BenchmarkClassObject(NULL),
ItascaClassObject(NULL, "OO1-ABSTRACT-CONNECTION") {
classObjectCtor("OO1-ABSTRACT-CONNECTION");
}

};
```

```
ItascaClassSetIter(Oo1AbstractConnectionClassObjectSetIterator, Oo1AbstractConnectionClassObject);


class Oo1AbstractConnection :  public virtual Benchmark {
public:
// Override functions (handled in-line via macros)
ItascaOverrideCopyObject(Oo1AbstractConnection)
ItascaOverrideGenericVersion(Oo1AbstractConnection)
ItascaOverrideMakeVersion(Oo1AbstractConnection)
ItascaOverrideObjectVersion(Oo1AbstractConnection)
ItascaOverrideParentVersion(Oo1AbstractConnection)
ItascaOverrideChildVersions(Oo1AbstractConnection)

// Select functions (handled in-line via macros)
ItascaOverrideSelectAnySimple(Oo1AbstractConnection,"OO1-ABSTRACT-CONNECTION")
ItascaOverrideSelectAnyStar(Oo1AbstractConnection,"OO1-ABSTRACT-CONNECTION")
ItascaOverrideSelectSimple(Oo1AbstractConnection,"OO1-ABSTRACT-CONNECTION")
ItascaOverrideSelectStar(Oo1AbstractConnection,"OO1-ABSTRACT-CONNECTION")

// Attributes and Methods

// Constructor which creates a new instance in ITASCA DB.
Oo1AbstractConnection()
: Benchmark(NULL),
ItascaObject(NULL,"OO1-ABSTRACT-CONNECTION") {
itascaObjectCtor("OO1-ABSTRACT-CONNECTION");
}

// Constructor which does NOT create a new instance.
Oo1AbstractConnection(const ItascaOID &new_oid)
: Benchmark(new_oid),
ItascaObject(new_oid,"OO1-ABSTRACT-CONNECTION") {
itascaObjectOidCtor(new_oid,"OO1-ABSTRACT-CONNECTION");
}

// Copy constructor.
Oo1AbstractConnection(const Oo1AbstractConnection &src_obj)
: Benchmark(NULL),
ItascaObject(NULL,"OO1-ABSTRACT-CONNECTION") {
objStatus = src_obj.baseCopyObject(itasca_oid);
};
};

ItascaClassSetIter(Oo1AbstractConnectionSetIterator, Oo1AbstractConnection);


class Oo1PartClassObject :  public virtual BenchmarkClassObject {
public:

// Attributes and Methods

// Constructors
Oo1PartClassObject(const ItascaOID &new_oid)
: BenchmarkClassObject(new_oid),
ItascaClassObject(new_oid, "OO1-PART") {
classObjectOidCtor(new_oid, "OO1-PART");
}
Oo1PartClassObject()
: BenchmarkClassObject(NULL),
ItascaClassObject(NULL, "OO1-PART") {
classObjectCtor("OO1-PART");
}
```

```
};

ItascaClassSetIter(Oo1PartClassObjectSetIterator, Oo1PartClassObject);



ItascaDeclareSetAttribute(Oo1AbstractConnection, Oo1AbstractConnectionSetAttribute);

class Oo1Part :  public virtual Benchmark {
public:
// Override functions (handled in-line via macros)
ItascaOverrideCopyObject(Oo1Part)
ItascaOverrideGenericVersion(Oo1Part)
ItascaOverrideMakeVersion(Oo1Part)
ItascaOverrideObjectVersion(Oo1Part)
ItascaOverrideParentVersion(Oo1Part)
ItascaOverrideChildVersions(Oo1Part)

// Select functions (handled in-line via macros)
ItascaOverrideSelectAnySimple(Oo1Part,"OO1-PART")
ItascaOverrideSelectAnyStar(Oo1Part,"OO1-PART")
ItascaOverrideSelectSimple(Oo1Part,"OO1-PART")
ItascaOverrideSelectStar(Oo1Part,"OO1-PART")

// Attributes and Methods
Oo1AbstractConnectionSetAttribute connectedTo;
// Inverse of \
Oo1AbstractConnectionSetAttribute connectedFrom;
// Inverse of \
ItascaInt build;
ItascaInt y;
ItascaInt x;
ItascaString type;
ItascaInt id;

// Constructor which creates a new instance in ITASCA DB.
Oo1Part()
: Benchmark(NULL),
ItascaObject(NULL,"OO1-PART"),
connectedTo("CONNECTED-TO", (ItascaObject *)this),
connectedFrom("CONNECTED-FROM", (ItascaObject *)this),
build("BUILD", (ItascaObject *)this),
y("Y", (ItascaObject *)this),
x("X", (ItascaObject *)this),
type("TYPE", (ItascaObject *)this),
id("ID", (ItascaObject *)this) {
itascaObjectCtor("OO1-PART");
}

// Constructor which does NOT create a new instance.
Oo1Part(const ItascaOID &new_oid)
: Benchmark(new_oid),
ItascaObject(new_oid,"OO1-PART"),
connectedTo("CONNECTED-TO", (ItascaObject *)this),
connectedFrom("CONNECTED-FROM", (ItascaObject *)this),
build("BUILD", (ItascaObject *)this),
y("Y", (ItascaObject *)this),
x("X", (ItascaObject *)this),
type("TYPE", (ItascaObject *)this),
id("ID", (ItascaObject *)this) {
itascaObjectOidCtor(new_oid,"OO1-PART");
}

// Copy constructor.
```

```
Oo1Part(const Oo1Part &src_obj)
: Benchmark(NULL),
ItascaObject(NULL,"OO1-PART") {
objStatus = src_obj.baseCopyObject(itasca_oid);
};
#include"part.hh"
};

ItascaClassSetIter(Oo1PartSetIterator, Oo1Part);


class Oo1ConnectionClassObject :  public virtual Oo1AbstractConnectionClassObject {
public:

// Attributes and Methods

// Constructors
Oo1ConnectionClassObject(const ItascaOID &new_oid)
: Oo1AbstractConnectionClassObject(new_oid),
BenchmarkClassObject(new_oid),
ItascaClassObject(new_oid, "OO1-CONNECTION") {
classObjectOidCtor(new_oid, "OO1-CONNECTION");
}
Oo1ConnectionClassObject()
: Oo1AbstractConnectionClassObject(NULL),
BenchmarkClassObject(NULL),
ItascaClassObject(NULL, "OO1-CONNECTION") {
classObjectCtor("OO1-CONNECTION");
}

};

ItascaClassSetIter(Oo1ConnectionClassObjectSetIterator, Oo1ConnectionClassObject);



ItascaDeclareClassNameAttribute(Oo1Part, Oo1PartAttribute);

class Oo1Connection :  public virtual Oo1AbstractConnection {
public:
// Override functions (handled in-line via macros)
ItascaOverrideCopyObject(Oo1Connection)
ItascaOverrideGenericVersion(Oo1Connection)
ItascaOverrideMakeVersion(Oo1Connection)
ItascaOverrideObjectVersion(Oo1Connection)
ItascaOverrideParentVersion(Oo1Connection)
ItascaOverrideChildVersions(Oo1Connection)

// Select functions (handled in-line via macros)
ItascaOverrideSelectAnySimple(Oo1Connection,"OO1-CONNECTION")
ItascaOverrideSelectAnyStar(Oo1Connection,"OO1-CONNECTION")
ItascaOverrideSelectSimple(Oo1Connection,"OO1-CONNECTION")
ItascaOverrideSelectStar(Oo1Connection,"OO1-CONNECTION")

// Attributes and Methods
Oo1PartAttribute to;
// Inverse of \
Oo1PartAttribute from;
// Inverse of \
ItascaInt cLength;
ItascaString type;

// Constructor which creates a new instance in ITASCA DB.
Oo1Connection()
```

```
: Oo1AbstractConnection(NULL),
Benchmark(NULL),
ItascaObject(NULL,"OO1-CONNECTION"),
to("TO", (ItascaObject *)this),
from("FROM", (ItascaObject *)this),
cLength("C-LENGTH", (ItascaObject *)this),
type("TYPE", (ItascaObject *)this) {
itascaObjectCtor("OO1-CONNECTION");
}

// Constructor which does NOT create a new instance.
Oo1Connection(const ItascaOID &new_oid)
: Oo1AbstractConnection(new_oid),
Benchmark(new_oid),
ItascaObject(new_oid,"OO1-CONNECTION"),
to("TO", (ItascaObject *)this),
from("FROM", (ItascaObject *)this),
cLength("C-LENGTH", (ItascaObject *)this),
type("TYPE", (ItascaObject *)this) {
itascaObjectOidCtor(new_oid,"OO1-CONNECTION");
}

// Copy constructor.
Oo1Connection(const Oo1Connection &src_obj)
: Oo1AbstractConnection(NULL),
Benchmark(NULL),
ItascaObject(NULL,"OO1-CONNECTION") {
objStatus = src_obj.baseCopyObject(itasca_oid);
};
#include"connect.hh"
};

ItascaClassSetIter(Oo1ConnectionSetIterator, Oo1Connection);


/* User defined keywords */

ItascaImplementSetAttribute(Oo1AbstractConnection, Oo1AbstractConnectionSetAttribute);

//
// end of file schema.hh
//
```

2.1.13 dbschema.dmp

The source code for the file *dbschema.dmp* is listed below. This file contains Itasca code, in the Lisp programming language, which creates the schema for the OO1 benchmark in the Itasca database. This file must be run on the Itasca database before the *bench* program is executed. The file must also be consistent with the *schema.hh* file, which describes the database schema for C++ programs which use it.

```
(in-package "ITASCA-USER")
(move-to-private-db 3)
(def-class BENCHMARK
 :document "top level object for Benchmark group"
 :superclasses (CLASS )
 :abstract T
 :versionable NIL
 :notify NIL
 :methods NIL
 :class-methods NIL
 :attributes ( )
 :class-attributes ( ))
```

```
(def-class 001-ABSTRACT-CONNECTION
 :document "Abstract connection class for the 001 benchmark."
 :superclasses (BENCHMARK )
 :abstract T
 :versionable NIL
 :notify NIL
 :methods NIL
 :class-methods NIL
 :attributes ( )
 :class-attributes ( ))

(def-class 001-CONNECTION
 :document NIL
 :superclasses (001-ABSTRACT-CONNECTION )
 :abstract NIL
 :versionable NIL
 :notify NIL
 :methods NIL
 :class-methods NIL
 :attributes ( (TYPE  :inherit-from NIL
 :document NIL
 :domain STRING
 :init ""
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 (C-LENGTH  :inherit-from NIL
 :document NIL
 :domain INTEGER
 :init 0
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 )
 :class-attributes ( ))

(def-class 001-PART
 :document "Part class for the 001 benchmark."
 :superclasses (BENCHMARK )
 :abstract NIL
 :versionable NIL
 :notify NIL
 :methods NIL
 :class-methods NIL
 :attributes ( (ID  :inherit-from NIL
 :document NIL
 :domain INTEGER
 :init 0
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
```

```
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 (TYPE  :inherit-from NIL
 :document NIL
 :domain STRING
 :init ""
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 (X  :inherit-from NIL
 :document NIL
 :domain INTEGER
 :init 0
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 (Y  :inherit-from NIL
 :document NIL
 :domain INTEGER
 :init 0
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 (BUILD  :inherit-from NIL
 :document NIL
 :domain INTEGER
 :init 0
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null T
)
 )
 :class-attributes ( ))


(change-attribute '001-CONNECTION 'FROM  :inherit-from 'NIL
 :document "Inverse of \"connected to\""
 :domain '001-PART
 :init 'NIL
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
```

```
 :unique-value NIL
 :unique-value* NIL
 :no-null NIL
)


(change-attribute 'OO1-CONNECTION 'TO  :inherit-from 'NIL
 :document "Inverse of \"connected from\""
 :domain 'OO1-PART
 :init 'NIL
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null NIL
)


(change-attribute 'OO1-PART 'CONNECTED-FROM  :inherit-from 'NIL
 :document "Inverse of \"to\""
 :domain '(SET-OF OO1-ABSTRACT-CONNECTION)
 :init 'NIL
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null NIL
)


(change-attribute 'OO1-PART 'CONNECTED-TO  :inherit-from 'NIL
 :document "Inverse of \"from\""
 :domain '(SET-OF OO1-ABSTRACT-CONNECTION)
 :init 'NIL
 :share NIL
 :composite NIL
 :exclusive NIL
 :dependent NIL
 :unique-value NIL
 :unique-value* NIL
 :no-null NIL
)
```

2.2 Matisse Implementation

This section lists the source code for the Matisse implementation of the OO1 benchmark created for this research. The implementation was created on version 2.2.0 of Matisse using the Object-Oriented Services API. The Matisse OO1 benchmark source code builds two executable programs, the *bench* program and the *oo1schema* program. The source files in this implementation are shown in Figure 2. The files *debug.hh*, *dice.hh*, *dice.cc*, *pmmlcg.h*, *pmmlcg.c*, *stopwtch.hh*, and *stopwtch.cc* are not listed, because they were described (with a full source code listing) in [2]. The following subsections list the source code for the other files shown in Figure 2.

2.2.1 Makefile

The *Makefile* for the Matisse OO1 benchmark source code is listed below.

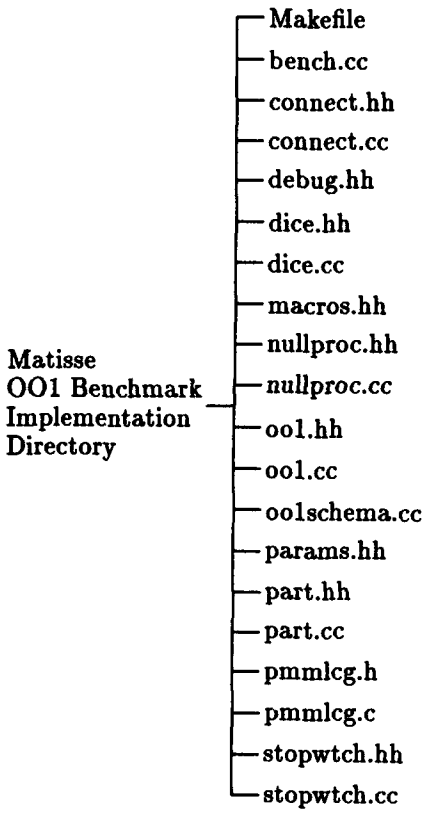

Figure 2: Matisse OO1 Benchmark Source Code Files

```
MATISSE_PATH=$(MTS_ROOT)
CCC=CC
#########################################################################
#
# If debug output is desired uncomment the "CCFLAGS" definition with
#  "-DDEBUG" in it.  Only one definition should be uncommented.
#
#CCFLAGS=-DDEBUG -g
CCFLAGS=
#
#########################################################################
LDFLAGS=-L$(MATISSE_PATH)/lib -Bstatic
INCLUDES=-I. -I$(MATISSE_PATH)/include
DE_LIBS=-lmatisseDE -lbsdmalloc
DS_LIBS=-lmatisseDS -lbsdmalloc

all: bench oo1schema

.c.o:
        $(CCC) $(CCFLAGS) $(INCLUDES) -c $<

.cc.o:
        $(CCC) $(CCFLAGS) $(INCLUDES) -c $<

pmmlcg.o: pmmlcg.c pmmlcg.h

dice.o: dice.cc debug.hh dice.hh pmmlcg.h

stopwtch.o: stopwtch.cc debug.hh stopwtch.hh

nullproc.o: nullproc.cc dice.hh

part.o: part.cc debug.hh macros.hh nullproc.hh oo1.hh params.hh part.hh

connect.o: connect.cc connect.hh debug.hh macros.hh oo1.hh

oo1.o: oo1.cc connect.hh debug.hh dice.hh macros.hh nullproc.hh
  oo1.hh params.hh part.hh stopwtch.hh

bench.o: bench.cc debug.hh macros.hh oo1.hh

bench: bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o
        $(CCC) $(LDFLAGS) -o bench \
         bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o \
         $(DE_LIBS)

oo1schema.o: oo1schema.cc debug.hh macros.hh

oo1schema: oo1schema.o
        $(CCC) $(LDFLAGS) -o oo1schema \
         oo1schema.o \
         $(DS_LIBS)

clean:
        rm -f *.o oo1schema bench
```

2.2.2 bench.cc

The source code for the file *bench.cc* is listed below. This file provides a driver for the benchmark. It consists of a main program which connects to the Matisse database, then performs the benchmark operation specified on the command line.

```
/*
```

```
 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 bench.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 16 Jul 1993

 This program performs all the benchmark measurements for the 001 benchmark.
  The following benchmark operations are supported: database creation (and
  load), lookup, forward traversal, reverse traversal, insert, and database
  clear (and destroy).

 Revisions:
 22 Aug 1993 -TJH- Changed out random number generator.
*/
#include<matisse.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"oo1.hh"
#include"macros.hh"

#define TRUE  1
#define FALSE 0
////////////////////////////////////////////////////////////////////////////
main( int argc, char **argv)
{
  DEBUG_INIT( "MAIN" , "main" );
  DEBUG_OUT( "entering", 1 );
  MtDatabase db;

  // check the command line arguments
  if ( argc < 6) {
    fprintf( stderr, "bench [operation] [server] [database] "
      "[# of parts] [random stream]\n" );
    return 1;  // exit the program
  }
  char *operation = argv[1];
  if ( !( !strcmp( operation, "load" ) || !strcmp( operation, "lookup" ) ||
    !strcmp( operation, "ftrav" ) || !strcmp( operation, "rtrav" ) ||
    !strcmp( operation, "insert" ) || !strcmp( operation, "clear" ) ) ) {
    fprintf( stderr, "ERROR[main] the operation must be \"load\", \"lookup\", "
      "\"ftrav\", \"rtrav\", \"insert\", or \"clear\"\n" );
    return 1;  // exit the program
  }
  char *db_server = argv[2];
  char *db_name = argv[3];
  int num_parts = atoi( argv[4] );
  if ( ( num_parts != 20000 ) && ( num_parts != 200000 ) )
    fprintf( stderr, "WARNING[main] the number of parts is not 20,000 or 200,000\n" );
  int stream = atoi( argv[5] );
```

```
  if ( ( stream < 1 ) || ( stream > 100 ) ) {
    fprintf( stderr, "ERROR[main] the stream is not between 1 and 100\n" );
    return 1;  // exit the program
  }

  // output a program title
  printf( "001 BENCHMARK [MATISSE] Air Force Institute of Technology\n" );
  printf( " MATISSE database \"%s\" on %s with %d parts (random stream %d).\n",
    db_name, db_server, num_parts, stream );

  // connect to the MATISSE database
  CHECK_STATUS( MtConnect( &db, db_server, db_name,
    3 /* priority */,
    5 /* wait */,
    FALSE /* no setTimeOnly */ ),
    "ERROR[main] unable to connect to MATISSE database" );

  // set the context (so that only one database is selected)
  CHECK_STATUS( MtSetContext( db ),
    "ERROR[main] unable to set context to opened database" );

  // create a single 001 benchmark object
  oo1 oo1_benchmark( num_parts, stream );

  // perform the benchmark measurement requested on the command line
  if ( !strcmp( operation, "load" ) ) {
    printf( " LOAD\n" );
    oo1_benchmark.load( );
  }
  else if ( !strcmp( operation, "lookup" ) ) {
    printf( " LOOKUP\n" );
    oo1_benchmark.lookup_measure( );
  }
  else if ( !strcmp( operation, "ftrav" ) ) {
    printf( " FORWARD TRAVERSAL\n" );
    oo1_benchmark.forward_traversal_measure( );
  }
  else if ( !strcmp( operation, "rtrav" ) ) {
    printf( " REVERSE TRAVERSAL\n" );
    oo1_benchmark.reverse_traversal_measure( );
  }
  else if ( !strcmp( operation, "insert" ) ) {
    printf( " INSERT\n" );
    oo1_benchmark.insert_measure( );
  }
  else if ( !strcmp( operation, "clear" ) ) {
    printf( " CLEAR\n" );
    oo1_benchmark.clear( );
  }

  // reset the context and close the database
  CHECK_STATUS( MtNoContext( ),
    "ERROR[main] unable to reset context" );
  CHECK_STATUS( MtDisconnect( db ),
    "ERROR[main] unable to close the MATISSE database" );
}
//////////////////////////////////////////////////////////////////////////
```

2.2.3 connect.hh

The source code for the file *connect.hh* is listed below. This file defines the non-persistent methods to be used with the *connection* class.

```
#ifndef __CONNECT_HH
```

```
#define __CONNECT_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 02 Aug 1993
*/
void new_connection( MtKey new_from, MtKey new_to, char *new_type, int new_length );
#endif __CONNECT_HH
```

2.2.4 connect.cc

The source code for the file *connect.cc* is listed below. This file implements the non-persistent methods for the *connection* class.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 02 Aug 1993

 Revisions:
 16 Sep 1993 -TJH- Changed use of several API functions to Mt_* functions,
                   (which use OIDs, rather than character strings) from
                   Mt* functions.  This was recommended by MATISSE to
                   improve performance.
*/
#include<matisse.h>
#include<stdlib.h>
#include<debug.hh>
#include"connect.hh"
#include"ooi.hh"
#include"macros.hh"
////////////////////////////////////////////////////////////////////////
void new_connection( MtKey new_from, MtKey new_to, char *new_type, int new_length )
```

```
{
  DEBUG_INIT( "CONNECTION" , "new_connection" );
  DEBUG_OUT( "entering", 1 );
  MtKey new_connection;

  CHECK_STATUS( Mt_CreateObject( &new_connection, oo1::connection_class ),
    "ERROR[new_connection] unable to create a connection instance" );
  CHECK_STATUS( Mt_SetValue( new_connection, oo1::connection_type, MTSTRING ,
    new_type, NULL ),
    "ERROR[new_connection] unable to set the connection type" );
  CHECK_STATUS( Mt_SetValue( new_connection, oo1::connection_length, MTS32,
    &new_length, NULL ),
    "ERROR[new_connection] unable to set the connection length" );
  CHECK_STATUS( Mt_AddSuccessor( new_connection, oo1::connection_to,
    new_to, MTAPPEND ),
    "ERROR[new_connection] unable to set the to relationship" );
  CHECK_STATUS( Mt_AddSuccessor( new_connection, oo1::connection_from,
    new_from, MTAPPEND ),
    "ERROR[new_connection] unable to set the from relationship" );
}
////////////////////////////////////////////////////////////////////////
```

2.2.5 macros.hh

The source code for the file *macros.hh* is listed below. This file contains the *CHECK_STATUS* macro which is used in all portions of the benchmark implementation to ensure that a database error did not occur during a call to the Matisse database.

```
#ifndef __MACROS_HH
#define __MACROS_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 macros.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 16 Jul 1993
*/
#define CHECK_STATUS(status,message)  \
  if (MtFailure (status)) {           \
    MtPError(message);                \
  exit(1); }
#endif __MACROS_HH
```

2.2.6 nullproc.hh

The source code for the file *nullproc.hh* is listed below. This file defines the null procedures which are required to be called at certain points in the OO1 benchmark.

```
#ifndef __NULLPROC_HH
```

```
#define __NULLPROC_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                   "null_procedure".
*/
void null_procedure( int x, int y, char *type );
void null_procedure_get_x_y( int& new_x, int& new_y, int stream );
#endif __NULLPROC_HH
```

2.2.7 nullproc.cc

The source code for the file *nullproc.cc* is listed below. This file implements the null procedures which are required to be called at certain points in the OO1 benchmark.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                   "null_procedure".
*/
#include<stdio.h>
#include<dice.hh>

#define TRUE  1
#define FALSE 0
```

```
static int debug = FALSE;
////////////////////////////////////////////////////////////////////////
void null_procedure( int x, int y, char *type )
{
  // this procedure does nothing (a null procedure)
  // (a special debug scheme is used here so that even a very smart
  //  compiler will still make this procedure call)
  if (debug)
    printf( "[null_procedure] %5d %5d %s\n", x, y, type );
}
////////////////////////////////////////////////////////////////////////
void null_procedure_get_x_y( int& new_x, int& new_y, int stream )
{
  new_x = (int)roll( 0, 99999, stream );
  new_y = (int)roll( 0, 99999, stream );
}
////////////////////////////////////////////////////////////////////////
```

2.2.8 oo1.hh

The source code for the file *oo1.hh* is listed below. This file defines the *OO1* class. This class implements the benchmark measures and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
#ifndef __OO1_HH
#define __OO1_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 31 Jul 1993
*/
#include"params.hh"

struct benchmark_results_type {
  double elapsed_time;
  double normalized_elapsed_time;
  int num_parts_connected;
} ;

class oo1 {
  int num_nodes_visited_in_forward_traversal;
  int num_parts;
  benchmark_results_type results[NUM_BENCHMARK_ITERATIONS];
  int stream;  // for the random number generator
  MtKey create_a_part( int part_id );
  void create_connections( MtKey db_part, int part_id, int num_parts );
  void lookup_schema_oids( );
  void report_results( int measurement );
```

```
public:
  static MtKey connection_class, connection_type, connection_length;
  static MtKey connection_from, connection_to;
  static MtKey part_class, part_id, part_type, part_x, part_y, part_build;
  static MtKey part_connected_from, part_connected_to;
  oo1( int num_parts, int stream );
  void clear( );
  void forward_traversal_measure( );
  void insert_measure( );
  void load( );
  void lookup_measure( );
  void reverse_traversal_measure( );
};
#endif __OO1_HH
```

2.2.9 oo1.cc

The source code for the file *oo1.cc* is listed below. This file implements the *OO1* class. This class implements the benchmark measures and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 31 Jul 1993

 Revisions:
 22 Aug 1993 -TJH- Changed out random number generator.
 22 Aug 1993 -TJH- Changed the "report_results( )" function to provide more
                   concise output.
 04 Sep 1993 -TJH- Changed use of several API functions to Mt_* functions,
                   (which use OIDs, rather than character strings) from
                   Mt* functions.  This was recommended by MATISSE to
                   improve performance.
 16 Sep 1993 -TJH- Removed MATISSE schema creation and removal from this program
                   (bench) into another program (oo1schema).
*/
#include<matisse.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
extern "C" int strftime(...);  // not defined in the <time.h> header file
#include<dice.hh>
#include<stopwtch.hh>
#include<debug.hh>
#include"oo1.hh"
#include"connect.hh"
#include"part.hh"
```

```
#include"nullproc.hh"
#include"params.hh"
#include"macros.hh"

enum { INSERT, FORWARD_TRAVERSAL, LOOKUP, REVERSE_TRAVERSAL };
static char *part_types[10] = {
  "part-type0", "part-type1", "part-type2", "part-type3", "part-type4",
  "part-type5", "part-type6", "part-type7", "part-type8", "part-type9"
};
////////////////////////////////////////////////////////////////////////
oo1::oo1( int num_parts, int stream )
{
  DEBUG_INIT( "001" , "oo1::oo1" );
  DEBUG_OUT( "entering", 1 );

  // save the number of parts in the database
  this->num_parts = num_parts;

  // save the stream number for use with the random number generator
  this->stream = stream;

  // calculate the number of nodes which will be visited in a forward traversal
  // (this value will be used to normalize the measurements
  //  from the reverse traversal)
  int nodes_at_current_level = NUM_CONNECTIONS;
  num_nodes_visited_in_forward_traversal = 1 + NUM_CONNECTIONS;
  for ( int i = 2 ; i <= TRAVERSAL_DEPTH ; i++ ) {
    nodes_at_current_level *= NUM_CONNECTIONS;
    num_nodes_visited_in_forward_traversal += nodes_at_current_level;
  }
  DEBUG_OUT( sprintf( BUG, "number of nodes which will be visited in the "
    "forward traversal %d", num_nodes_visited_in_forward_traversal ), 2 );

  // determine all the database schema oids
  lookup_schema_oids( );
  DEBUG_OUT( "MATISSE OIDs read in", 2 );
}
////////////////////////////////////////////////////////////////////////
void oo1::clear( )
{
  DEBUG_INIT( "001" , "oo1::clear" );
  DEBUG_OUT( "entering", 1 );

  // clear all the parts from the database
  CHECK_STATUS( MtStartTransaction( 0 ),
    "ERROR[oo1::clear] unable to start a transaction" );

  int num_parts_to_delete = num_parts;
  while ( num_parts_to_delete ) {
    delete_part( num_parts_to_delete );
    num_parts_to_delete--;
  }

  CHECK_STATUS( MtCommitTransaction( 0, 0 ),
    "ERROR[oo1::clear] unable to commit current transaction" );

  // note that this procedure doesn't delete the schema
  DEBUG_OUT( "all parts and connections deleted (but the schema remains)", 2 );
}
////////////////////////////////////////////////////////////////////////
MtKey oo1::create_a_part( int part_id )  // private
{
  DEBUG_INIT( "001" , "oo1::create_a_part" );
  DEBUG_OUT( "entering", 1 );
```

```
  int new_x;
  int new_y;
  char text_build_date[10];

  // create a new part in the database
  char *new_type = part_types[roll( 0, 9, stream )];
  null_procedure_get_x_y( new_x, new_y, stream );  // null procedure
  long new_build_datetime = roll( JAN_1_1980, JAN_1_1990, stream );
  struct tm *new_tm_timedate = localtime( &new_build_datetime );
  strftime( text_build_date, 10, "%d%b%Y", new_tm_timedate );
  DEBUG_OUT( sprintf( BUG, "adding part %d (%s %5d %5d %s)",
    part_id, new_type, new_x, new_y, text_build_date ), 2 );
  return ( new_part( part_id, new_type, new_x, new_y, new_build_datetime ) );
}
////////////////////////////////////////////////////////////////////////
void oo1::create_connections( MtKey db_part, int part_id, int num_parts )  // private
{
  DEBUG_INIT( "001" , "oo1::create_connections" );
  DEBUG_OUT( "entering", 1 );
  int cpart_id;
  MtKey db_cpart;
  int new_length;
  char *new_type;

  // create NUM_CONNECTIONS from the part "db_part"
  for ( int c = 1 ; c <= NUM_CONNECTIONS ; c++ ) {
    if ( roll( 1, 10, stream ) > LOCALITY_OF_REFERENCE ) {
      // 90% of the time create connection to the closest 1% of parts
      //  (this is true when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = part_id + (int)roll( 1, (long)( num_parts / 100 ), stream )
        - 1 - (int)( num_parts / 200 );
      // "double up" at the ends (to stay in the part id range)
      if ( cpart_id < (int)( num_parts / 200 ) )
        cpart_id = cpart_id + (int)( num_parts / 200 );
      if ( cpart_id > ( num_parts - (int)( num_parts / 200 ) ) )
        cpart_id = cpart_id - (int)( num_parts / 200 );
    }
    else {
      // 10% of the time create connection to any part 1..num_parts
      //  (this is true when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = (int)roll( 1, (long)num_parts, stream );
    }

    // create the connection in the database
    db_cpart = query_a_part( cpart_id );  // query the part
    new_type = part_types[roll( 0, 9, stream )];
    new_length = (int)roll( 0, 99999, stream );
    DEBUG_OUT( sprintf( BUG, "connecting part %d to part %d (%s %5d)",
      part_id, cpart_id, new_type, new_length ), 2 );
    new_connection( db_part, db_cpart, new_type, new_length );
  }
}
////////////////////////////////////////////////////////////////////////
void oo1::forward_traversal_measure( )
{
  DEBUG_INIT( "001" , "oo1::forward_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  MtKey db_part;
  int part_id;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
```

```
    stopwatch traversal_timer;
    traversal_timer.start( );

    CHECK_STATUS( MtStartTransaction( 0 ),
      "ERROR[oo1::forward_traversal_measure] unable to start a transaction" );

    for ( int j = 1 ; j <= NUM_FORWARD_TRAVERSAL ; j++ ) {

      // lookup a random part
      part_id = (int)roll( 1, (long)num_parts, stream );
      db_part = query_a_part( part_id );  // lookup the part

      // find all the parts connected to this part (up to TRAVERSAL_DEPTH)
      results[i - 1].num_parts_connected = forward_traversal( 0, db_part );
      DEBUG_OUT( sprintf( BUG, "found %d parts connected to part %d",
        results[i - 1].num_parts_connected, part_id ), 2 );
    }

    CHECK_STATUS( MtCommitTransaction( 0, 0 ),
      "ERROR[oo1::forward_traversal_measure] unable to commit current transaction" );

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = traversal_timer.stop( );
  }

  // report the benchmark results
  report_results( FORWARD_TRAVERSAL );
}
//////////////////////////////////////////////////////////////////////
void oo1::insert_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::insert_measure" );
  DEBUG_OUT( "entering", 1 );

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch insert_timer;
    insert_timer.start( );

    CHECK_STATUS( MtStartTransaction( 0 ),
      "ERROR[oo1::insert_measure] unable to start a transaction" );

    // insert parts into the database and call a "null" procedure
    //  to get the x and y position for each insert
    // (the "null" procedure is called by the "create_a_part( )" function)
    for ( int p = num_parts + 1 ; p <= num_parts + NUM_INSERT ; p++ )
      create_connections( create_a_part( p ), p, p );

    CHECK_STATUS( MtCommitTransaction( 0, 0 ),
      "ERROR[oo1::insert_measure] unable to commit current transaction" );

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = insert_timer.stop( );

    // remove the inserted parts and connections
    CHECK_STATUS( MtStartTransaction( 0 ),
      "ERROR[oo1::insert_measure] unable to start a transaction" );

    for ( p = num_parts + 1 ; p <= num_parts + NUM_INSERT ; p++ )
      delete_part( p );

    CHECK_STATUS( MtCommitTransaction( 0, 0 ),
```

```
      "ERROR[oo1::insert_measure] unable to commit current transaction" );
  }

  // report the benchmark results
  report_results( INSERT );
}
////////////////////////////////////////////////////////////////////////
void oo1::load( )
{
  DEBUG_INIT( "OO1" , "oo1::load" );
  DEBUG_OUT( "entering", 1 );

  // time the creation of the OO1 database
  stopwatch db_creation_timer;
  db_creation_timer.start( );

  CHECK_STATUS( MtStartTransaction( 0 ),
    "ERROR[oo1::load] unable to start a transaction" );

  // create "num_part" parts
  for ( int p = 1 ; p <= num_parts ; p++ )
    create_a_part( p );

  // create connections for each part
  MtKey db_part;
  for ( p = 1 ; p <= num_parts ; p++ ) {
    db_part = query_a_part( p );
    create_connections( db_part, p, num_parts );
  }

  CHECK_STATUS( MtCommitTransaction( 0, 0 ),
    "ERROR[oo1::load] unable to commit current transaction" );

  // report the time it took to load the database
  double total_elapsed_time = db_creation_timer.stop( );
  printf( " %.3f sec\n", total_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::lookup_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::lookup_measure" );
  DEBUG_OUT( "entering", 1 );
  MtKey db_part;
  MtS32 db_part_id;
  int part_id_to_lookup;
  MtChar part_type_char[11];
  MtSize size;
  MtType type;
  int x;
  int y;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch lookup_timer;
    lookup_timer.start( );

    CHECK_STATUS( MtStartTransaction( 0 ),
      "ERROR[oo1::lookup_measure] unable to start a transaction" );

    // lookup parts in the database and call a null procedure for each lookup
    for ( int j = 1 ; j <= NUM_LOOKUP ; j++ ) {
```

```
      // lookup a random part
      part_id_to_lookup = (int)roll( 1, (long)num_parts, stream );
      db_part = query_a_part( part_id_to_lookup );  // lookup the part

      // read the part's "id"
      size = sizeof( db_part_id );
      CHECK_STATUS( Mt_GetValue( db_part, oo1::part_id, &type, &db_part_id, NULL,
        &size, NULL ),
        "ERROR[oo1::lookup_measure] unable to get value of part id" );
      // read the part's "type"
      size = sizeof( part_type_char );
      CHECK_STATUS( Mt_GetValue( db_part, oo1::part_type, &type, part_type_char, NULL,
        &size, NULL ),
        "ERROR[oo1::lookup_measure] unable to get value of part type" );
      // read the part's "x" and "y" values
      size = sizeof( x );
      CHECK_STATUS( Mt_GetValue( db_part, oo1::part_x, &type, &x, NULL, &size, NULL ),
        "ERROR[oo1::lookup_measure] unable to get value of part x" );
      size = sizeof( y );
      CHECK_STATUS( Mt_GetValue( db_part, oo1::part_y, &type, &y, NULL, &size, NULL ),
        "ERROR[oo1::lookup_measure] unable to get value of part y" );

      DEBUG_OUT( sprintf( BUG, "looked up part %d", db_part_id ), 2 );

      // call the required null procedure
      null_procedure( x, y, part_type_char );
    }

    CHECK_STATUS( MtCommitTransaction( 0, 0 ),
      "ERROR[oo1::lookup_measure] unable to commit current transaction" );

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = lookup_timer.stop( );
  }

  // report the benchmark results
  report_results( LOOKUP );
}
////////////////////////////////////////////////////////////////////////
void oo1::lookup_schema_oids( )  // private
{
  DEBUG_INIT( "001" , "oo1::lookup_schema_oids" );
  DEBUG_OUT( "entering", 1 );

  CHECK_STATUS( MtStartTransaction( 0 ),
    "ERROR[oo1::lookup_schema_oids] unable to start a transaction" );

  // lookup the schema for the part class
  CHECK_STATUS( MtGetClass( &part_class, "part" ),
    "Error[oo1::lookup_schema_oids] can't find part class" );
  CHECK_STATUS( MtGetAttribute( &part_id, "part id" ),
    "Error[oo1::lookup_schema_oids] can't find part id attribute" );
  CHECK_STATUS( MtGetAttribute( &part_type, "part type" ),
    "Error[oo1::lookup_schema_oids] can't find part type attribute" );
  CHECK_STATUS( MtGetAttribute( &part_x, "part x" ),
    "Error[oo1::lookup_schema_oids] can't find part x attribute" );
  CHECK_STATUS( MtGetAttribute( &part_y, "part y" ),
    "Error[oo1::lookup_schema_oids] can't find part y attribute" );
  CHECK_STATUS( MtGetAttribute( &part_build, "part build" ),
    "Error[oo1::lookup_schema_oids] can't find part build attribute" );
  CHECK_STATUS( MtGetRelationship( &part_connected_to, "part connected to" ),
    "Error[oo1::lookup_schema_oids] can't find part connected to relationship" );
  CHECK_STATUS( MtGetRelationship( &part_connected_from, "part connected from" ),
    "Error[oo1::lookup_schema_oids] can't find part connected from relationship" );
```

```
  // lookup the schema for the connection class
  CHECK_STATUS( MtGetClass( &connection_class, "connection" ),
    "Error[oo1::lookup_schema_oids] can't find connection class" );
  CHECK_STATUS( MtGetAttribute( &connection_type, "connection type" ),
    "Error[oo1::lookup_schema_oids] can't find connection type attribute" );
  CHECK_STATUS( MtGetAttribute( &connection_length, "connection length" ),
    "Error[oo1::lookup_schema_oids] can't find connection length attribute" );
  CHECK_STATUS( MtGetRelationship( &connection_to, "connection to" ),
    "Error[oo1::lookup_schema_oids] can't find connection to relationship" );
  CHECK_STATUS( MtGetRelationship( &connection_from, "connection from" ),
    "Error[oo1::lookup_schema_oids] can't find connection from relationship" );

  CHECK_STATUS( MtCommitTransaction( 0, 0 ),
    "ERROR[oo1::lookup_schema_oids] unable to commit current transaction" );
}
////////////////////////////////////////////////////////////////////////
void oo1::report_results( int measurement )  // private
{
  { // two debug INITs used in this function so hide this one from the other
    DEBUG_INIT( "OO1" , "oo1::report_results" );
    DEBUG_OUT( "entering", 1 );
  }
  double cold_elapsed_time;
  double total_after_first_iteration = 0.0;
  double warm_elapsed_time;

  // report the benchmark results
  DEBUG_INIT( "RESULTS" , "oo1::report_results" );

  // determine the cold time
  if ( measurement == REVERSE_TRAVERSAL )
    // the reverse traversal times are normalized
    cold_elapsed_time = results[0].normalized_elapsed_time;
  else
    cold_elapsed_time = results[0].elapsed_time;

  // calculate the warm time
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // output detailed results if the "RESULTS" environment variable is set
    //  to at least a level of 1 (i.e. setenv RESULTS 1)
    DEBUG_OUT( sprintf( BUG, "iteration %2d elapsed time %.3f sec", i,
      results[i - 1].elapsed_time ), 1 );
    if ( measurement == FORWARD_TRAVERSAL ) {
      DEBUG_OUT( sprintf( BUG, "(parts found %d)",
        results[i - 1].num_parts_connected ), 1 );
    }
    if ( measurement == REVERSE_TRAVERSAL ) {
      DEBUG_OUT( sprintf( BUG, "(parts found %d normalized time %.3f sec)",
        results[i - 1].num_parts_connected,
        results[i - 1].normalized_elapsed_time ), 1 );
    }

    // add up the elapsed times (for all iterations after the first)
    if (i != 1)
      if ( measurement == REVERSE_TRAVERSAL )
        total_after_first_iteration += results[i - 1].normalized_elapsed_time;
      else
        total_after_first_iteration += results[i - 1].elapsed_time;
  }
  warm_elapsed_time = total_after_first_iteration / ( NUM_BENCHMARK_ITERATIONS - 1 );

  // output a quick summary of all the times
```

```
  for ( i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {
    printf( " %7.3f", results[i - 1].elapsed_time );
  }
  // output the cold and warm results
  printf( " C %7.3f", cold_elapsed_time );
  printf( " W %7.3f\n", warm_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::reverse_traversal_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::reverse_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  MtKey db_part;
  int part_id;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch traversal_timer;
    traversal_timer.start( );

    CHECK_STATUS( MtStartTransaction( 0 ),
      "ERROR[oo1::reverse_traversal_measure] unable to start a transaction" );

    for ( int j = 1 ; j <= NUM_FORWARD_TRAVERSAL ; j++ ) {

      // lookup a random part
      part_id = (int)roll( 1, (long)num_parts, stream );
      db_part = query_a_part( part_id );  // lookup the part

      // find all the parts which connect to this part (up to TRAVERSAL_DEPTH)
      results[i - 1].num_parts_connected = reverse_traversal( 0, db_part );
      DEBUG_OUT( sprintf( BUG, "found %d parts connected to part %d",
        results[i - 1].num_parts_connected, part_id ), 2 );
    }

    CHECK_STATUS( MtCommitTransaction( 0, 0 ),
      "ERROR[oo1::reverse_traversal_measure] unable to commit current transaction" );

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = traversal_timer.stop( );
    // the results for the reverse traversal are normalized so that they
    //  may be compared to the forward traversal
    // (multiply the time by num_nodes_visited_in_forward_traversal/N,
    //  where N is the number of nodes actually visited in the reverse
    //  traversal)
    results[i - 1].normalized_elapsed_time = results[i - 1].elapsed_time *
      ( (double)num_nodes_visited_in_forward_traversal /
      (double)results[i - 1].num_parts_connected );
  }

  // report the benchmark results
  report_results( REVERSE_TRAVERSAL );
}
////////////////////////////////////////////////////////////////////////
```

2.2.10 oo1schema.cc

The source code for the file *oo1schema.cc* is listed below. This file is the source file for the *oo1schema* program. The *oo1shema* program loads (and can remove) the OO1 benchmark schema into the Matisse database. The schema must be loaded before the *bench* program can be run.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1schema.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 16 Sep 1993

 This program creates or removes the MATISSE schema for the 001 benchmark.
  This program requires linking to the "matisseDS" library, while the "bench"
  program only requires linking to the "matisseDE" library (because it does
  not modify the database schema).  The use of the DE library for the "bench"
  program was recommended by MATISSE to improve database performance.
*/
#include<matisse.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"macros.hh"

void create_schema( );
void remove_schema( );

#define TRUE  1
#define FALSE 0
#define SCHEMA_ERROR "ERROR[create_schema] error while creating schema"
////////////////////////////////////////////////////////////////////////////
main( int argc, char **argv)
{
  DEBUG_INIT( "MAIN" , "main" );
  DEBUG_OUT( "entering", 1 );
  MtDatabase db;

  // check the command line arguments
  if ( argc < 4) {
    fprintf( stderr, "oo1schema [create|remove] [server] [database]\n" );
    return 1;  // exit the program
  }
  char *operation = argv[1];
  if ( !( !strcmp( operation, "create" ) || !strcmp( operation, "remove" ) ) ) {
    fprintf( stderr, "ERROR[main] the operation must be \"create\", or "
      "\"remove\"\n" );
    return 1;  // exit the program
  }
  char *db_server = argv[2];
  char *db_name = argv[3];

  // output a program title
  printf( "001 BENCHMARK [MATISSE] Air Force Institute of Technology\n" );
  printf( " MATISSE database \"%s\" on %s.\n", db_name, db_server );
```

```
  // connect to the MATISSE database
  CHECK_STATUS( MtConnect( &db, db_server, db_name,
    3 /* priority */,
    5 /* wait */,
    FALSE /* no setTimeOnly */ ),
    "ERROR[main] unable to connect to MATISSE database" );

  // set the context (so that only one database is selected)
  CHECK_STATUS( MtSetContext( db ),
    "ERROR[main] unable to set context to opened database" );

  // create or delete the benchmark schema
  if ( !strcmp( operation, "create" ) ) {
    printf( " SCHEMA CREATE\n" );
    create_schema( );
  }
  else if ( !strcmp( operation, "remove" ) ) {
    printf( " SCHEMA REMOVE\n" );
    remove_schema( );
  }

  // reset the context and close the database
  CHECK_STATUS( MtNoContext( ),
    "ERROR[main] unable to reset context" );
  CHECK_STATUS( MtDisconnect( db ),
    "ERROR[main] unable to close the MATISSE database" );
}
////////////////////////////////////////////////////////////////////////
void create_schema( )
{
  DEBUG_INIT( "MAIN" , "create_schema" );
  DEBUG_OUT( "entering", 1 );
  MtKey connection_class, connection_type, connection_length;
  MtKey connection_from, connection_to;
  MtType int_type[2];
  MtKey part_class, part_id, part_type, part_x, part_y, part_build;
  MtKey part_connected_from, part_connected_to;
  int part_cardinality[2];
  MtType string_type[1];

  int_type[0] = MTNIL;
  int_type[1] = MTS32;
  part_cardinality[0] = 0;
  part_cardinality[1] = -1;
  string_type[0] = MTSTRING;

  CHECK_STATUS( MtStartTransaction( 0 ),
    "ERROR[create_schema] unable to start a transaction" );

  // create classes
  CHECK_STATUS( MtCreateObject( &part_class, "class" ),
    SCHEMA_ERROR );
  CHECK_STATUS( MtSetValue( part_class, "external name", MTSTRING, "part", 0 ),
    SCHEMA_ERROR );

  CHECK_STATUS( MtCreateObject( &connection_class, "class" ),
    SCHEMA_ERROR );
  CHECK_STATUS( MtSetValue( connection_class, "external name", MTSTRING,
    "connection", 0 ),
    SCHEMA_ERROR );

  // create attributes
  CHECK_STATUS( MtCreateObject( &part_id, "attribute" ),
    SCHEMA_ERROR );
```

```
CHECK_STATUS( MtSetValue( part_id, "external name", MTSTRING, "part id", 0 ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_id, "make entry function", MTSTRING,
  "make-entry", 0 ), SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_id, "type", MTS32_LIST, int_type, 1, 2 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &part_type, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_type, "external name", MTSTRING, "part type", 0 ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_type, "type", MTS32_LIST, string_type, 1, 1 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &part_x, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_x, "external name", MTSTRING, "part x", 0 ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_x, "type", MTS32_LIST, int_type, 1, 2 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &part_y, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_y, "external name", MTSTRING, "part y", 0 ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_y, "type", MTS32_LIST, int_type, 1, 2 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &part_build, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_build, "external name", MTSTRING, "part build", 0 ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_build, "type", MTS32_LIST, int_type, 1, 2 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &connection_type, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( connection_type, "external name", MTSTRING,
  "connection type", 0 ), SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( connection_type, "type", MTS32_LIST, string_type, 1, 1 ),
  SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &connection_length, "attribute" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( connection_length, "external name", MTSTRING,
  "connection length", 0 ), SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( connection_length, "type", MTS32_LIST, int_type, 1, 2 ),
  SCHEMA_ERROR );

// add relationships
CHECK_STATUS( MtCreateObject( &part_connected_to, "relationship" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_connected_to, "external name", MTSTRING,
  "part connected to", 0 ), SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( part_connected_to, "cardinality", MTS32_LIST,
  part_cardinality, 1, 2 ), SCHEMA_ERROR );
CHECK_STATUS( MtAddSuccessor( part_connected_to, "successors",
  connection_class,MTAPPEND ), SCHEMA_ERROR );

CHECK_STATUS( MtCreateObject( &connection_from, "relationship" ),
  SCHEMA_ERROR );
CHECK_STATUS( MtSetValue( connection_from, "external name", MTSTRING,
  "connection from", 0 ), SCHEMA_ERROR );
CHECK_STATUS( MtAddSuccessor( connection_from, "successors", part_class, MTAPPEND ),
```

```
    SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessor( connection_from, "inverse relationship",
    part_connected_to, MTAPPEND ), SCHEMA_ERROR );

  CHECK_STATUS( MtCreateObject( &part_connected_from, "relationship" ),
    SCHEMA_ERROR );
  CHECK_STATUS( MtSetValue( part_connected_from, "external name", MTSTRING,
    "part connected from", 0 ), SCHEMA_ERROR );
  CHECK_STATUS( MtSetValue( part_connected_from, "cardinality", MTS32_LIST,
    part_cardinality, 1, 2 ), SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessor( part_connected_from, "successors",
    connection_class, MTAPPEND ), SCHEMA_ERROR );

  CHECK_STATUS( MtCreateObject( &connection_to, "relationship" ),
    SCHEMA_ERROR );
  CHECK_STATUS( MtSetValue( connection_to, "external name", MTSTRING,
    "connection to", 0 ), SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessor( connection_to, "successors", part_class, MTAPPEND ),
    SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessor( connection_to, "inverse relationship",
    part_connected_from, MTAPPEND ), SCHEMA_ERROR );

  // add successors
  CHECK_STATUS( MtAddSuccessors( part_class, "attributes", 5, part_id,
    part_type, part_x, part_y, part_build ), SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessors( part_class, "relationships", 2,
    part_connected_to, part_connected_from ), SCHEMA_ERROR );

  CHECK_STATUS( MtAddSuccessors( connection_class, "attributes", 2,
    connection_type, connection_length ), SCHEMA_ERROR );
  CHECK_STATUS( MtAddSuccessors( connection_class, "relationships", 2,
    connection_from, connection_to ), SCHEMA_ERROR );

  CHECK_STATUS( MtCommitTransaction( 0, 0 ),
    "ERROR[create_schema] unable to commit current transaction" );
}
////////////////////////////////////////////////////////////////////////
void remove_schema( )
{
  DEBUG_INIT( "MAIN" , "remove_schema" );
  DEBUG_OUT( "entering", 1 );
  MtKey connection_class, connection_type, connection_length;
  MtKey connection_from, connection_to;
  MtKey part_class, part_id, part_type, part_x, part_y, part_build;
  MtKey part_connected_from, part_connected_to;

  CHECK_STATUS( MtStartTransaction( 0 ),
    "ERROR[remove_schema] unable to start a transaction" );

  // lookup the schema for the part class
  CHECK_STATUS( MtGetClass( &part_class, "part" ),
    "Error[remove_schema] can't find part class" );
  CHECK_STATUS( MtGetAttribute( &part_id, "part id" ),
    "Error[remove_schema] can't find part id attribute" );
  CHECK_STATUS( MtGetAttribute( &part_type, "part type" ),
    "Error[remove_schema] can't find part type attribute" );
  CHECK_STATUS( MtGetAttribute( &part_x, "part x" ),
    "Error[remove_schema] can't find part x attribute" );
  CHECK_STATUS( MtGetAttribute( &part_y, "part y" ),
    "Error[remove_schema] can't find part y attribute" );
  CHECK_STATUS( MtGetAttribute( &part_build, "part build" ),
    "Error[remove_schema] can't find part build attribute" );
  CHECK_STATUS( MtGetRelationship( &part_connected_to, "part connected to" ),
    "Error[remove_schema] can't find part connected to relationship" );
```

```
  CHECK_STATUS( MtGetRelationship( &part_connected_from, "part connected from" ),
    "Error[remove_schema] can't find part connected from relationship" );

  // lookup the schema for the connection class
  CHECK_STATUS( MtGetClass( &connection_class, "connection" ),
    "Error[remove_schema] can't find connection class" );
  CHECK_STATUS( MtGetAttribute( &connection_type, "connection type" ),
    "Error[remove_schema] can't find connection type attribute" );
  CHECK_STATUS( MtGetAttribute( &connection_length, "connection length" ),
    "Error[remove_schema] can't find connection length attribute" );
  CHECK_STATUS( MtGetRelationship( &connection_to, "connection to" ),
    "Error[remove_schema] can't find connection to relationship" );
  CHECK_STATUS( MtGetRelationship( &connection_from, "connection from" ),
    "Error[remove_schema] can't find connection from relationship" );

  // remove the part class schema
  CHECK_STATUS( MtRemoveObject( part_class ),
    "Error[remove_schema] can't remove part class" );
  CHECK_STATUS( MtRemoveObject( part_id ),
    "Error[remove_schema] can't remove part id attribute" );
  CHECK_STATUS( MtRemoveObject( part_type ),
    "Error[remove_schema] can't remove part type attribute" );
  CHECK_STATUS( MtRemoveObject( part_x ),
    "Error[remove_schema] can't remove part x attribute" );
  CHECK_STATUS( MtRemoveObject( part_y ),
    "Error[remove_schema] can't remove part y attribute" );
  CHECK_STATUS( MtRemoveObject( part_build ),
    "Error[remove_schema] can't remove part build attribute" );
  CHECK_STATUS( MtRemoveObject( part_connected_to ),
    "Error[remove_schema] can't remove part connected to relationship" );
  CHECK_STATUS( MtRemoveObject( part_connected_from ),
    "Error[remove_schema] can't remove part connected from relationship" );

  // remove the connection class schema
  CHECK_STATUS( MtRemoveObject( connection_class ),
    "Error[remove_schema] can't remove connection class" );
  CHECK_STATUS( MtRemoveObject( connection_type ),
    "Error[remove_schema] can't remove connection type attribute" );
  CHECK_STATUS( MtRemoveObject( connection_length ),
    "Error[remove_schema] can't remove connection length attribute" );
  CHECK_STATUS( MtRemoveObject( connection_to ),
    "Error[remove_schema] can't remove connection to relationship" );
  CHECK_STATUS( MtRemoveObject( connection_from ),
    "Error[remove_schema] can't remove connection from relationship" );

  CHECK_STATUS( MtCommitTransaction( 0, 0 ),
    "ERROR[remove_schema] unable to commit current transaction" );
}
//////////////////////////////////////////////////////////////////////
```

2.2.11 params.hh

The source code for the file *params.hh* is listed below. This file defines several constant parameters which the OO1 benchmark implementation requires.

```
#ifndef __PARAMS_HH
#define __PARAMS_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
```

```
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 params.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 31 Jul 1993
*/
// definition of the number of connections for each part (3 for 001)
#define NUM_CONNECTIONS 3
// definition of the depth of the forward and reverse traversals (7 for 001)
#define TRAVERSAL_DEPTH 7
// definition of the locality of reference value (1 for 001 which means that
//  90% of the connections are randomly selected among the 1% of the parts
//  which are closest (in terms of part id), and the rest of the connections
//  are made to any randomly selected part.
#define LOCALITY_OF_REFERENCE 1
// definitions to create a 10 year range of dates
#define JAN_1_1980 315550800
#define JAN_1_1990 631170000
// 001 benchmark parameters
#define NUM_BENCHMARK_ITERATIONS 10
#define NUM_LOOKUP 1000
#define NUM_FORWARD_TRAVERSAL 1
#define NUM_REVERSE_TRAVERSAL 1
#define NUM_INSERT 100
#endif __PARAMS_HH
```

2.2.12 part.hh

The source code for the file *part.hh* is listed below. This file defines the non-persistent methods to be used with the *part* class.

```
#ifndef __PART_HH
#define __PART_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 part.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 02 Aug 1993
*/
MtKey new_part( int new_id, char *new_type, int new_x, int new_y, long new_build );
void delete_part( int part_id );
```

```
MtKey query_a_part( int part_id );
int forward_traversal( int current_level, MtKey db_part );
int reverse_traversal( int current_level, MtKey db_part );
#endif __PART_HH
```

2.2.13 part.cc

The source code for the file *part.cc* is listed below. This file implements the non-persistent methods for the *part* class.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    MATISSE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 part.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 02 Aug 1993

 Revisions:
 16 Sep 1993 -TJH- Changed use of several API functions to Mt_* functions,
                   (which use OIDs, rather than character strings) from
                   Mt* functions.  This was recommended by MATISSE to
                   improve performance.
*/
#include<matisse.h>
#include<stdlib.h>
#include<debug.hh>
#include"part.hh"
#include"oo1.hh"
#include"nullproc.hh"
#include"params.hh"
#include"macros.hh"
////////////////////////////////////////////////////////////////////////
MtKey new_part( int new_id, char *new_type, int new_x, int new_y, long new_build )
{
  DEBUG_INIT( "PART" , "new_part" );
  DEBUG_OUT( "entering", 1 );
  MtKey new_part;

  CHECK_STATUS( Mt_CreateObject( &new_part, oo1::part_class ),
    "ERROR[new_part] unable to create a part instance" );
  CHECK_STATUS( Mt_SetValue( new_part, oo1::part_id, MTS32, &new_id, NULL ),
    "ERROR[new_part] unable to set the part id" );
  CHECK_STATUS( Mt_SetValue( new_part, oo1::part_type, MTSTRING , new_type, NULL ),
    "ERROR[new_part] unable to set the part id" );
  CHECK_STATUS( Mt_SetValue( new_part, oo1::part_x, MTS32, &new_x, NULL ),
    "ERROR[new_part] unable to set the part x" );
  CHECK_STATUS( Mt_SetValue( new_part, oo1::part_y, MTS32, &new_y, NULL ),
    "ERROR[new_part] unable to set the part y" );
  CHECK_STATUS( Mt_SetValue( new_part, oo1::part_build, MTS32, &new_build, NULL ),
    "ERROR[new_part] unable to set the part build" );
```

```
  return ( new_part );
}
////////////////////////////////////////////////////////////////////////
void delete_part( int part_id )
{
  DEBUG_INIT( "PART" , "delete_part" );
  DEBUG_OUT( "entering", 1 );
  MtKey a_connection;
  MtStream connection_stream;
  MtKey connections_to_delete[100]; // hope this is large enough!
  int num_connections_to_delete;
  MtSize num_of_parts;
  char part_id_string[20];
  MtKey *table_of_parts;

  // query the part which is going to be deleted
  // (don't use the "query_a_part" function because if there is more than
  //  one (or zero) we still want the parts deleted)
  sprintf( part_id_string, "%d", part_id );
  CHECK_STATUS( Mt_MGetObjectsFromEP( &num_of_parts, &table_of_parts,
    part_id_string, oo1::part_id, oo1::part_class ),
    "ERROR[delete_part] object get from entry point failed" );;

  while (num_of_parts-- ) {

    // need to remove all connections to and from this part to keep the
    //  database consistant

    // delete all the connections from this part to other parts
    num_connections_to_delete = 0;
    CHECK_STATUS( Mt_OpenRelStream( &connection_stream,
      table_of_parts[num_of_parts], oo1::part_connected_to ),
      "ERROR[delete_part] unable to open a stream on the connections" );
    while ( MtNextObject( connection_stream, &a_connection ) == MATISSE_SUCCESS) {
      connections_to_delete[num_connections_to_delete] = a_connection;
      num_connections_to_delete++;
      DEBUG_OUT( "found a connection to", 2 );
    }
    CHECK_STATUS( MtCloseStream( connection_stream ),
      "ERROR[delete_part] can't close the connection stream" );
    // delete all the connections found (the inverse relationship will take
    //  care of the other side of the relationship)
    while (num_connections_to_delete--) {
      CHECK_STATUS(
        MtRemoveObject( connections_to_delete[num_connections_to_delete] ),
        "ERROR[delete_part] unable to remove a connection object" );
    }

    // delete all the connections to this part from other parts
    num_connections_to_delete = 0;
    CHECK_STATUS( Mt_OpenRelStream( &connection_stream,
      table_of_parts[num_of_parts], oo1::part_connected_from ),
      "ERROR[delete_part] unable to open a stream on the connections" );
    while ( MtNextObject( connection_stream, &a_connection ) == MATISSE_SUCCESS) {
      connections_to_delete[num_connections_to_delete] = a_connection;
      num_connections_to_delete++;
      DEBUG_OUT( "found a connection from", 2 );
    }
    CHECK_STATUS( MtCloseStream( connection_stream ),
      "ERROR[delete_part] can't close the connection stream" );
    // delete all the connections found (the inverse relationship will take
    //  care of the other side of the relationship)
    while (num_connections_to_delete--) {
```

```
      CHECK_STATUS(
        MtRemoveObject( connections_to_delete[num_connections_to_delete] ),
        "ERROR[delete_part] unable to remove a connection object" );
    }

    // remove the part
    CHECK_STATUS( MtRemoveObject( table_of_parts[num_of_parts] ),
      "ERROR[delete_part] unable to remove an part object" );
  }
  free( table_of_parts );
}
////////////////////////////////////////////////////////////////////////
MtKey query_a_part( int part_id )
{
  DEBUG_INIT( "PART" , "query_a_part" );
  DEBUG_OUT( "entering", 1 );
  MtKey *table_of_parts;
  MtSize num_of_parts;
  char part_id_string[20];
  MtKey return_part_id;

  // query the part
  sprintf( part_id_string, "%d", part_id );
  CHECK_STATUS( Mt_MGetObjectsFromEP( &num_of_parts, &table_of_parts,
    part_id_string, oo1::part_id, oo1::part_class ),
    "ERROR[query_a_part] MATISSE failed query for a part" );
  if ( num_of_parts != 1 ) {
    MtPError( "ERROR[query_a_part] found none (or to many) parts" );
  }
  return_part_id = table_of_parts[0];
  free( table_of_parts );

  return ( return_part_id );
}
////////////////////////////////////////////////////////////////////////
int forward_traversal( int current_level, MtKey db_part )
{
  DEBUG_INIT( "PART" , "forward_traversal" );
  DEBUG_OUT( "entering", 1 );
  MtKey a_connection;
  MtStream connection_stream;
  MtS32 db_part_id;
  MtSize num_of_parts;
  MtChar part_type[11];
  MtSize size;
  MtKey *table_of_parts;
  MtType type;
  int x;
  int y;

  // read the part's "id"
  size = sizeof( db_part_id );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_id, &type, &db_part_id, NULL,
    &size, NULL ),
    "ERROR[forward_traversal] unable to get value of part id" );
  // read the part's "type"
  size = sizeof( part_type );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_type, &type, part_type, NULL,
    &size, NULL ),
    "ERROR[forward_traversal] unable to get value of part type" );
  // read the part's "x" and "y" values
  size = sizeof( x );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_x, &type, &x, NULL, &size, NULL ),
    "ERROR[forward_traversal] unable to get value of part x" );
```

```
  size = sizeof( y );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_y, &type, &y, NULL, &size, NULL ),
    "ERROR[forward_traversal] unable to get value of part y" );

  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)", db_part_id, current_level ), 2 );

  // call the required null procedure
  null_procedure( x, y, part_type );

  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {

    CHECK_STATUS( Mt_OpenRelStream( &connection_stream, db_part,
      oo1::part_connected_to ),
      "ERROR[forward_traversal] unable to open a stream on the connections" );

    while ( MtNextObject( connection_stream, &a_connection ) == MATISSE_SUCCESS ) {

      CHECK_STATUS( Mt_MGetSuccessors( &num_of_parts, &table_of_parts,
        a_connection, oo1::connection_to ),
        "ERROR[forward_traversal] unable find successors of connection to" );

      if ( num_of_parts != 1 ) {
        MtPError( "ERROR[forward_traversal] found none (or to many) parts" );
      }
      num_parts_connected +=
        forward_traversal( current_level + 1, table_of_parts[0] );
    }
    CHECK_STATUS( MtCloseStream( connection_stream ),
      "ERROR[forward_traversal] can't close the connection stream" );
  }
  return ( num_parts_connected + 1 );
}
//////////////////////////////////////////////////////////////////////////
int reverse_traversal( int current_level, MtKey db_part )
{
  DEBUG_INIT( "PART" , "reverse_traversal" );
  DEBUG_OUT( "entering", 1 );
  MtKey a_connection;
  MtStream connection_stream;
  MtS32 db_part_id;
  MtSize num_of_parts;
  MtChar part_type[11];
  MtSize size;
  MtKey *table_of_parts;
  MtType type;
  int x;
  int y;

  // read the part's "id"
  size = sizeof( db_part_id );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_id, &type, &db_part_id, NULL,
    &size, NULL ),
    "ERROR[reverse_traversal] unable to get value of part id" );
  // read the part's "type"
  size = sizeof( part_type );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_type, &type, part_type, NULL,
    &size, NULL ),
    "ERROR[reverse_traversal] unable to get value of part type" );
  // read the part's "x" and "y" values
  size = sizeof( x );
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_x, &type, &x, NULL, &size, NULL ),
    "ERROR[reverse_traversal] unable to get value of part x" );
  size = sizeof( y );
```

```
  CHECK_STATUS( Mt_GetValue( db_part, oo1::part_y, &type, &y, NULL, &size, NULL ),
    "ERROR[reverse_traversal] unable to get value of part y" );

  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)", db_part_id, current_level ), 2 );

  // call the required null procedure
  null_procedure( x, y, part_type );

  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {

    CHECK_STATUS( Mt_OpenRelStream( &connection_stream, db_part,
      oo1::part_connected_from ),
      "ERROR[reverse_traversal] unable to open a stream on the connections" );

    while ( MtNextObject( connection_stream, &a_connection ) == MATISSE_SUCCESS ) {

      CHECK_STATUS( Mt_MGetSuccessors( &num_of_parts, &table_of_parts,
        a_connection, oo1::connection_from ),
        "ERROR[reverse_traversal] unable find successors of connection from" );

      if ( num_of_parts != 1 ) {
        MtPError( "ERROR[reverse_traversal] found none (or to many) parts" );
      }
      num_parts_connected +=
        reverse_traversal( current_level + 1, table_of_parts[0] );
    }
    CHECK_STATUS( MtCloseStream( connection_stream ),
      "ERROR[reverse_traversal] can't close the connection stream" );
  }
  return ( num_parts_connected + 1 );
}
//////////////////////////////////////////////////////////////////////
```

2.3 ObjectStore Implementation

This section lists the source code for the ObjectStore implementation of the OO1 benchmark created for this research. The implementation was created on version 2.0.1 of ObjectStore using the C++ DML API. The ObjectStore OO1 benchmark source code builds one executable program, the *bench* program. The source files in this implementation are shown in Figure 3. The files *debug.hh*, *dice.hh*, *dice.cc*, *pmmlcg.h*, *pmmlcg.c*, *stopwtch.hh*, and *stopwtch.cc* are not listed, because they were described (with a full source code listing) in [2]. The following subsections list the source code for the other files shown in Figure 3.

2.3.1 Makefile

The *Makefile* for the ObjectStore OO1 benchmark source code is listed below.

```
include $(OS_ROOTDIR)/etc/ostore.mk
OS_COMPILATION_SCHEMA_DB_PATH=
  prowler:/usr3/databases/ostore_data/oo1.comp_schema
OS_APPLICATION_SCHEMA_DB_PATH=
  prowler:/usr3/databases/ostore_data/oo1.app_schema
#########################################################################
#
# If debug output is desired uncomment the "CCFLAGS" definition with
#  "-DDEBUG" in it.  Only one definition should be uncommented.
#
#CCFLAGS=-I. -DDEBUG -gx
CCFLAGS=-I. -gx
#
#########################################################################
```

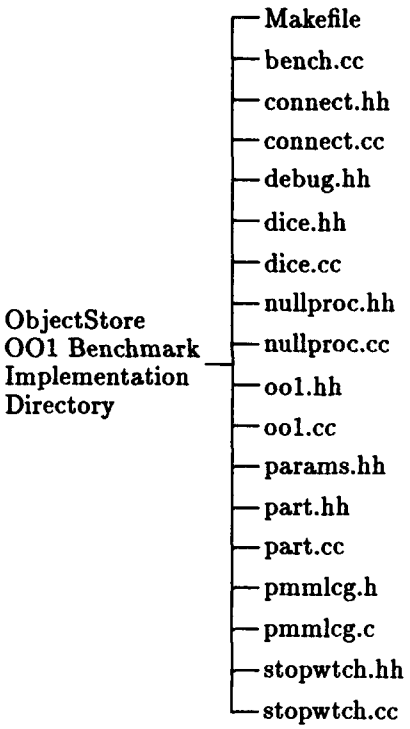

Figure 3: ObjectStore OO1 Benchmark Source Code Files

```
LDFLAGS=-g
LDLIBS=-loscol -los
SOURCES=bench.cc connect.cc nullproc.cc oo1.cc part.cc stopwtch.cc
  dice.cc pmmlcg.c
OBJECTS=bench.o connect.o nullproc.o oo1.o part.o stopwtch.o dice.o pmmlcg.o
EXECUTABLES=bench

all: ${EXECUTABLES}

bench: bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o
        ${LINK.cc} -o bench \
          bench.o oo1.o part.o connect.o nullproc.o stopwtch.o dice.o pmmlcg.o \
          ${LDLIBS}

clean:
        rm -f ${EXECUTABLES} ${OBJECTS}
        osrm -f /usr3/databases/ostore_data/oo1.comp_schema

depend:
        osmakedep .depend $(CCFLAGS) $(CPPFLAGS) -files $(SOURCES)

include .depend
```

2.3.2 bench.cc

The source code for the file *bench.cc* is listed below. This file provides a driver for the benchmark. It consists of a main program which connects to the Itasca database, then performs the benchmark operation specified on the command line.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 bench.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 06 Jul 1993

 This program performs all the benchmark measurements for the 001 benchmark.
  The following benchmark operations are supported: database creation (and
  load), lookup, forward traversal, reverse traversal, insert, and database
  clear (and destroy).

 Revisions:
 19 Jul 1993 -TJH- Added support to "clear" the benchmark database.
 18 Aug 1993 -TJH- Added code so that ObjectStore would not try to read the
                   entire segment at a time (bad for large database) and
                   also set the fetch policy to os_fetch_page.
 18 Aug 1993 -TJH- Changed out random number generator.
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
```

```
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<debug.hh>
#include"oo1.hh"

#define TRUE 1
#define FALSE 0

os_database *db = 0;  // required to be here by ObjectStore
////////////////////////////////////////////////////////////////////////
main( int argc, char **argv)
{
  DEBUG_INIT( "MAIN" , "main" );
  DEBUG_OUT( "entering", 1 );

  // check the command line arguments
  if ( argc < 5) {
    fprintf( stderr, "bench [operation] [database] "
      "[# of parts] [random stream]\n" );
    return 1;  // exit the program
  }
  char *operation = argv[1];
  if ( !( !strcmp( operation, "load" ) || !strcmp( operation, "lookup" ) ||
    !strcmp( operation, "ftrav" ) || !strcmp( operation, "rtrav" ) ||
    !strcmp( operation, "insert" ) || !strcmp( operation, "clear" ) ) ) {
    fprintf( stderr, "ERROR[main] the operation must be \"load\", \"lookup\", "
      "\"ftrav\", \"rtrav\", \"insert\", or \"clear\"\n" );
    return 1;  // exit the program
  }
  char *db_name = argv[2];
  int num_parts = atoi( argv[3] );
  if ( ( num_parts != 20000 ) && ( num_parts != 200000 ) )
    fprintf( stderr, "WARNING[main] the number of parts is not 20,000 or 200,000\n" );
  int stream = atoi( argv[4] );
  if ( ( stream < 1 ) || ( stream > 100 ) ) {
    fprintf( stderr, "ERROR[main] the stream is not between 1 and 100\n" );
    return 1;  // exit the program
  }

  // output a program title
  printf( "OO1 BENCHMARK [OBJECTSTORE] Air Force Institute of Technology\n" );
  printf( " ObjectStore database \"%s\" with %d parts (random stream %d).\n",
    db_name, num_parts, stream );

  // initialize ObjectStore
  objectstore::initialize( );
  os_collection::initialize( );

  // open (or create for "load" operation) the database
  if ( !strcmp( argv[1], "load" ) )
    db = os_database::create( db_name );
  else
    db = os_database::open( db_name );

  // tell ObjectStore not to read the entire segment
  db->set_read_whole_segment( FALSE );

  // set the fetch policy of ObjectStore to fetch a number of bytes at a time
  db->set_fetch_policy( os_fetch_page, FETCH_SIZE_IN_BYTES );

  // create a single OO1 benchmark object
  oo1 oo1_benchmark( num_parts, stream );
```

```
  // perform the benchmark measurement requested on the command line
  if ( !strcmp( operation, "load" ) ) {
    printf( " LOAD\n" );
    oo1_benchmark.load( );
  }
  else if ( !strcmp( operation, "lookup" ) ) {
    printf( " LOOKUP\n" );
    oo1_benchmark.lookup_measure( );
  }
  else if ( !strcmp( operation, "ftrav" ) ) {
    printf( " FORWARD TRAVERSAL\n" );
    oo1_benchmark.forward_traversal_measure( );
  }
  else if ( !strcmp( operation, "rtrav" ) ) {
    printf( " REVERSE TRAVERSAL\n" );
    oo1_benchmark.reverse_traversal_measure( );
  }
  else if ( !strcmp( operation, "insert" ) ) {
    printf( " INSERT\n" );
    oo1_benchmark.insert_measure( );
  }
  else if ( !strcmp( operation, "clear" ) ) {
    printf( " CLEAR\n" );
    oo1_benchmark.clear( );
  }
  db->close( );
}
////////////////////////////////////////////////////////////////////////////
```

2.3.3 connect.hh

The source code for the file *connect.hh* is listed below. This file defines the *connection* class.

```
#ifndef __CONNECT_HH
#define __CONNECT_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 18 Jun 1993
*/
extern os_database* db;
class part;

class connection {
public:
  part *from inverse_member connected_to;
  part *to inverse_member connected_from;
  char type[11];
  int length;
```

```
  connection( part *new_from, part *new_to, char *new_type, int new_length );
  ~connection( );
};
#endif __CONNECT_HH
```

2.3.4 connect.cc

The source code for the file *connect.cc* is listed below. This file implements the methods for the *connection* class.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 connect.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 06 Jul 1993 -TJH- Changed all the debug output to use the "debug.hh" macros.
 29 Jul 1993 -TJH- Added a destructor so that connection objects can delete
                   themselves.
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<debug.hh>
#include"part.hh"
#include"connect.hh"
////////////////////////////////////////////////////////////////////////
connection::connection( part *new_from, part *new_to, char *new_type,
  int new_length )
{
  DEBUG_INIT( "CONNECTION" , "connection::connection" );
  DEBUG_OUT( "entering", 1 );

  from = new_from;
  to = new_to;
  strcpy( type, new_type );
  length = new_length;
}
////////////////////////////////////////////////////////////////////////
connection::~connection( )
{
  DEBUG_INIT( "CONNECTION" , "connection::~connection" );
  DEBUG_OUT( "entering", 1 );

  from->connected_to.remove( this );
  to->connected_from.remove( this );
}
```

2.3.5 nullproc.hh

The source code for the file *nullproc.hh* is listed below. This file defines the null procedures which are required to be called at certain points in the OO1 benchmark.

```
#ifndef __NULLPROC_HH
#define __NULLPROC_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                   "null_procedure".
*/
void null_procedure( int x, int y, char *type );
void null_procedure_get_x_y( int& new_x, int& new_y, int stream );
#endif __NULLPROC_HH
```

2.3.6 nullproc.cc

The source code for the file *nullproc.cc* is listed below. This file implements the null procedures which are required to be called at certain points in the OO1 benchmark.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 nullproc.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 19 Jun 1993

 Revisions:
 01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                   "null_procedure".
```

```
*/
#include<stdio.h>
#include<dice.hh>

#define TRUE  1
#define FALSE 0
static int debug = FALSE;
//////////////////////////////////////////////////////////////////////////
void null_procedure( int x, int y, char *type )
{
  // this procedure does nothing (a null procedure)
  // (a special debug scheme is used here so that even a very smart
  //  compiler will still make this procedure call)
  if (debug)
    printf( "[null_procedure] %5d %5d %s\n", x, y, type );
}
//////////////////////////////////////////////////////////////////////////
void null_procedure_get_x_y( int& new_x, int& new_y, int stream )
{
  new_x = (int)roll( 0, 99999, stream );
  new_y = (int)roll( 0, 99999, stream );
}
//////////////////////////////////////////////////////////////////////////
```

2.3.7 oo1.hh

The source code for the file *oo1.hh* is listed below. This file defines the *OO1* class. This class implements the benchmark measures and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
#ifndef __OO1_HH
#define __OO1_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 06 Jul 1993

 Revisions:
 07 Jul 1993 -TJH- The variable "num_parts" was added as a class attribute,
                   and removed as a parameter to every class method.
*/
#include"connect.hh"
#include"part.hh"
#include"params.hh"

struct benchmark_results_type {
  double elapsed_time;
  double normalized_elapsed_time;
```

```
  int num_parts_connected;
} ;

class oo1 {
  int num_nodes_visited_in_forward_traversal;
  int num_parts;
  benchmark_results_type results[NUM_BENCHMARK_ITERATIONS];
  int stream;  // for the random number generator
  part *create_a_part( int part_id );
  void create_connections( part *db_part, int num_parts );
  void report_results( int measurement );
public:
  oo1( int num_parts, int stream );
  void clear( );
  void forward_traversal_measure( );
  void insert_measure( );
  void load( );
  void lookup_measure( );
  void reverse_traversal_measure( );
};
#endif __OO1_HH
```

2.3.8 oo1.cc

The source code for the file *oo1.cc* is listed below. This file implements the *OO1* class. This class implements the benchmark measures and reporting functions required by the OO1 benchmark. The *OO1* class is non-persistent.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 oo1.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 06 Jul 1993

 Revisions: (revisions before 06 Jul 1993 are from previous programs)
 01 Jul 1993 -TJH- Modified to include "params.hh" which contains the
                   definitions of all benchmark parameters.
 01 Jul 1993 -TJH- Modified the program so that elapsed time measurements for
                   each iteration are stored in an array (called "results").
                   The results are output after all the benchmark timing
                   is done.
 02 Jul 1993 -TJH- Changed the database build into a single transaction.
 02 Jul 1993 -TJH- Created an index on the "id" attribute of the "part"
                   class.  This optimizes part id lookups (by queries).
 06 Jul 1993 -TJH- Combined all the benchmark measurements into a single
                   class, the "oo1" class.  The programs "loaddb", "lookup",
                   "ftrav", "rtrav", and "insert" were merged.
 07 Jul 1993 -TJH- The variable "num_parts" was added as a class attribute,
                   and removed as a parameter to every class method.
```

```
 07 Jul 1993 -TJH- Moved starting and stoping the benchmark timer to outside
                   of the database transactions (i.e. outside the
                   "do_transaction( ) { }" statement).
 19 Jul 1993 -TJH- Combined functions so they could be used by "load" and
                   "insert" to reduce the total amount of code.
 19 Jul 1993 -TJH- Added the "clear" function to destroy the benchmark
                   database.
 20 Jul 1993 -TJH- Created the "report_results" function to reduce the
                   total amount of code.
 20 Jul 1993 -TJH- Fixed several calculations which were not using the
                   constants in "params.hh" properly (and would have
                   caused errors if the values were changed).
 29 Jul 1993 -TJH- Fixed a major bug in "oo1::insert_measure( )."  The insert
                   was not deleting the connection objects which were created.
                   This was causing "core" dumps on the reverse traversal,
                   which was finding these connection objects and trying to
                   follow them back to part objects (which no longer existed).
 02 Aug 1993 -TJH- Moved the removal of connections from a part which is being
                   deleted from the "oo1::insert_measure( )" function to the
                   destructor of the part class.
 18 Aug 1993 -TJH- Changed out random number generator.
 19 Aug 1993 -TJH- Changed the "report_results( )" function to provide more
                   concise output.
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
extern "C" int strftime(...);  // not defined in the <time.h> header file
#include<dice.hh>
#include<stopwtch.hh>
#include<debug.hh>
#include"oo1.hh"
#include"connect.hh"
#include"part.hh"
#include"nullproc.hh"
#include"params.hh"

enum { INSERT, FORWARD_TRAVERSAL, LOOKUP, REVERSE_TRAVERSAL };
static char *part_types[10] = {
  "part-type0", "part-type1", "part-type2", "part-type3", "part-type4",
  "part-type5", "part-type6", "part-type7", "part-type8", "part-type9"
};

persistent<db> os_Set<part*>*part::extent = 0;
extern os_database *db;
////////////////////////////////////////////////////////////////////////
oo1::oo1( int num_parts, int stream )
{
  DEBUG_INIT( "OO1" , "oo1::oo1" );
  DEBUG_OUT( "entering", 1 );

  // save the number of parts in the database
  this->num_parts = num_parts;

  // save the stream number for use with the random number generator
  this->stream = stream;

  // calculate the number of nodes which will be visited in a forward traversal
  // (this value will be used to normalize the measurements
  //  from the reverse traversal)
  int nodes_at_current_level = NUM_CONNECTIONS;
```

```
  num_nodes_visited_in_forward_traversal = 1 + NUM_CONNECTIONS;
  for ( int i = 2 ; i <= TRAVERSAL_DEPTH ; i++ ) {
    nodes_at_current_level *= NUM_CONNECTIONS;
    num_nodes_visited_in_forward_traversal += nodes_at_current_level;
  }
  DEBUG_OUT( sprintf( BUG, "number of nodes which will be visited in the "
    "forward traversal %d", num_nodes_visited_in_forward_traversal ), 2 );
}
////////////////////////////////////////////////////////////////////////
void oo1::clear( )
{
  DEBUG_INIT( "OO1" , "oo1::clear" );
  DEBUG_OUT( "entering", 1 );

  do_transaction( ) {
    db->destroy( );
    DEBUG_OUT( "database destroyed", 2 );
  }
}
////////////////////////////////////////////////////////////////////////
part *oo1::create_a_part( int part_id )  // private
{
  DEBUG_INIT( "OO1" , "oo1::create_a_part" );
  DEBUG_OUT( "entering", 1 );
  int new_x;
  int new_y;
  char text_build_date[10];

  // create a new part in the database
  char *new_type = part_types[roll( 0, 9, stream )];
  null_procedure_get_x_y( new_x, new_y, stream );  // null procedure
  long new_build_datetime = roll( JAN_1_1980, JAN_1_1990, stream );
  struct tm *new_tm_timedate = localtime( &new_build_datetime );
  strftime( text_build_date, 10, "%d%b%Y", new_tm_timedate );
  DEBUG_OUT( sprintf( BUG, "adding part %d (%s %5d %5d %s)",
    part_id, new_type, new_x, new_y, text_build_date ), 2 );
  return ( new(db) part( part_id, new_type, new_x, new_y, new_build_datetime ) );
}
////////////////////////////////////////////////////////////////////////
void oo1::create_connections( part *db_part, int num_parts )  // private
{
  DEBUG_INIT( "OO1" , "oo1::create_connections" );
  DEBUG_OUT( "entering", 1 );
  int cpart_id;
  part *db_cpart;
  connection *new_connection;
  int new_length;
  char *new_type;

  // create NUM_CONNECTIONS from the part "db_part"
  for ( int c = 1 ; c <= NUM_CONNECTIONS ; c++ ) {
    if ( roll( 1, 10, stream ) > LOCALITY_OF_REFERENCE ) {
      // 90% of the time create connection to the closest 1% of parts
      //  (this is tr e when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = db_part->id + (int)roll( 1, (long)( num_parts / 100 ), stream )
        - 1 - (int)( num_parts / 200 );
      // "double up" at the ends (to stay in the part id range)
      if ( cpart_id < (int)( num_parts / 200 ) )
        cpart_id = cpart_id + (int)( num_parts / 200 );
      if ( cpart_id > ( num_parts - (int)( num_parts / 200 ) ) )
        cpart_id = cpart_id - (int)( num_parts / 200 );
    }
    else {
      // 10% of the time create connection to any part 1..num_parts
```

```
      //  (this is true when "LOCALITY_OF_REFERENCE" is equal to 1)
      cpart_id = (int)roll( 1, (long)num_parts, stream );
    }

    // create the connection in the database
    db_cpart = ( *part::extent )[% id == cpart_id %];  // query the part
    if ( db_cpart->id != cpart_id )
      fprintf( stderr, "WARNING[oo1::create_connections] invalid query result\n" );
    new_type = part_types[roll( 0, 9, stream )];
    new_length = (int)roll( 0, 99999, stream );
    DEBUG_OUT( sprintf( BUG, "connecting part %d to part %d (%s %5d)",
      db_part->id, db_cpart->id, new_type, new_length ), 2 );
    new_connection = new(db) connection( db_part, db_cpart, new_type, new_length );
  }
}
//////////////////////////////////////////////////////////////////////
void oo1::forward_traversal_measure( )
{
  DEBUG_INIT( "001" , "oo1::forward_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  part *db_part;
  int part_id;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch traversal_timer;
    traversal_timer.start( );

    do_transaction( ) {
      for ( int j = 1 ; j <= NUM_FORWARD_TRAVERSAL ; j++ ) {

        // lookup a random part
        part_id = (int)roll( 1, (long)num_parts, stream );
        db_part = ( *part::extent )[% id == part_id %];  // lookup the part

        // find all the parts connected to this part (up to TRAVERSAL_DEPTH)
        results[i - 1].num_parts_connected = db_part->forward_traversal( 0 );
        DEBUG_OUT( sprintf( BUG, "found %d parts connected to part %d",
          results[i - 1].num_parts_connected, db_part->id ), 2 );
      }
    }

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = traversal_timer.stop( );
  }

  // report the benchmark results
  report_results( FORWARD_TRAVERSAL );
}
//////////////////////////////////////////////////////////////////////
void oo1::insert_measure( )
{
  DEBUG_INIT( "001" , "oo1::insert_measure" );
  DEBUG_OUT( "entering", 1 );

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch insert_timer;
    insert_timer.start( );
```

```
    do_transaction( ) {

      // insert parts into the database and call a "null" procedure
      //  to get the x and y position for each insert
      // (the "null" procedure is called by the "create_a_part( )" function)
      for ( int p = num_parts + 1 ; p <= num_parts + NUM_INSERT ; p++ )
        create_connections( create_a_part( p ), p );
    }

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = insert_timer.stop( );

    // remove the inserted parts and connections
    do_transaction( ) {
      int largest_original_part_id = num_parts;
      os_Set<part*> selected_parts =
        ( *part::extent )[: id > largest_original_part_id :];
      selected_parts.change_behavior( os_collection::maintain_cursors );
      foreach ( part *p, selected_parts, os_cursor::safe ) {
        DEBUG_OUT( sprintf( BUG, "deleting part %d", p->id ), 2);
        delete p;
      }
    }
  }

  // report the benchmark results
  report_results( INSERT );
}
////////////////////////////////////////////////////////////////////////
void oo1::load( )
{
  DEBUG_INIT( "OO1" , "oo1::load" );
  DEBUG_OUT( "entering", 1 );

  // time the creation of the OO1 database
  stopwatch db_creation_timer;
  db_creation_timer.start( );

  do_transaction( ) {

    // create an extent for the part class
    part::extent = &os_Set<part*>::create( db, 0,
      (os_int32)num_parts /* tell ObjectStore the expected size */ );

    // create "num_part" parts
    for ( int p = 1 ; p <= num_parts ; p++ )
      create_a_part( p );

    // create an index to optimize lookup by part id
    os_index_path &key_spec =
      os_index_path::create( "part*", "id", db );
    part::extent->add_index( key_spec );

    // create connections for each part
    foreach ( part *db_part, *part::extent )
      create_connections( db_part, num_parts );
  }

  // report the time it took to load the database
  double total_elapsed_time = db_creation_timer.stop( );
  printf( " %.3f sec\n", total_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::lookup_measure( )
```

```
{
  DEBUG_INIT( "OO1" , "oo1::lookup_measure" );
  DEBUG_OUT( "entering", 1 );
  part *db_part;
  int part_id;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch lookup_timer;
    lookup_timer.start( );

    do_transaction( ) {

      // lookup parts in the database and call a null procedure for each lookup
      for ( int j = 1 ; j <= NUM_LOOKUP ; j++ ) {

        // lookup a random part
        part_id = (int)roll( 1, (long)num_parts, stream );
        db_part = ( *part::extent )[% id == part_id %];  // lookup the part
        DEBUG_OUT( sprintf( BUG, "looked up part %d", db_part->id ), 2 );

        // call the required null procedure
        null_procedure( db_part->x, db_part->y, db_part->type );
      }
    }

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = lookup_timer.stop( );
  }

  // report the benchmark results
  report_results( LOOKUP );
}
////////////////////////////////////////////////////////////////////////
void oo1::report_results( int measurement )  // private
{
  { // two debug INITs used in this function so hide this one from the other
    DEBUG_INIT( "OO1" , "oo1::report_results" );
    DEBUG_OUT( "entering", 1 );
  }
  double cold_elapsed_time;
  double total_after_first_iteration = 0.0;
  double warm_elapsed_time;

  // report the benchmark results
  DEBUG_INIT( "RESULTS" , "oo1::report_results" );

  // determine the cold time
  if ( measurement == REVERSE_TRAVERSAL )
    // the reverse traversal times are normalized
    cold_elapsed_time = results[0].normalized_elapsed_time;
  else
    cold_elapsed_time = results[0].elapsed_time;

  // calculate the warm time
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // output detailed results if the "RESULTS" environment variable is set
    //  to at least a level of 1 (i.e. setenv RESULTS 1)
    DEBUG_OUT( sprintf( BUG, "iteration %2d elapsed time %.3f sec", i,
      results[i - 1].elapsed_time ), 1 );
    if ( measurement == FORWARD_TRAVERSAL ) {
```

```
      DEBUG_OUT( sprintf( BUG, "(parts found %d)",
        results[i - 1].num_parts_connected ), 1 );
    }
    if ( measurement == REVERSE_TRAVERSAL ) {
      DEBUG_OUT( sprintf( BUG, "(parts found %d normalized time %.3f sec)",
        results[i - 1].num_parts_connected,
        results[i - 1].normalized_elapsed_time ), 1 );
    }

    // add up the elapsed times (for all iterations after the first)
    if (i != 1)
      if ( measurement == REVERSE_TRAVERSAL )
        total_after_first_iteration += results[i - 1].normalized_elapsed_time;
      else
        total_after_first_iteration += results[i - 1].elapsed_time;
  }
  warm_elapsed_time = total_after_first_iteration / ( NUM_BENCHMARK_ITERATIONS - 1 );

  // output a quick summary of all the times
  for ( i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {
    printf( " %7.3f", results[i - 1].elapsed_time );
  }
  // output the cold and warm results
  printf( " C %7.3f", cold_elapsed_time );
  printf( " W %7.3f\n", warm_elapsed_time );
}
////////////////////////////////////////////////////////////////////////
void oo1::reverse_traversal_measure( )
{
  DEBUG_INIT( "OO1" , "oo1::reverse_traversal_measure" );
  DEBUG_OUT( "entering", 1 );
  part *db_part;
  int part_id;

  // run the benchmark iterations
  for ( int i = 1 ; i <= NUM_BENCHMARK_ITERATIONS ; i++ ) {

    // start the timer
    stopwatch traversal_timer;
    traversal_timer.start( );

    do_transaction( ) {
      for ( int j = 1 ; j <= NUM_REVERSE_TRAVERSAL ; j++ ) {

        // lookup a random part
        part_id = (int)roll( 1, (long)num_parts, stream );
        db_part = ( *part::extent )[% id == part_id %];  // lookup the part

        // find all the parts which connect to this part (up to TRAVERSAL_DEPTH)
        results[i - 1].num_parts_connected = db_part->reverse_traversal( 0 );
        DEBUG_OUT( sprintf( BUG, "found %d parts which connect to part %d",
          results[i - 1].num_parts_connected, db_part->id ), 2 );
      }
    }

    // stop the timer and save the elapsed time
    results[i - 1].elapsed_time = traversal_timer.stop( );
    // the results for the reverse traversal are normalized so that they
    //  may be compared to the forward traversal
    // (multiply the time by num_nodes_visited_in_forward_traversal/N,
    //  where N is the number of nodes actually visited in the reverse
    //  traversal)
    results[i - 1].normalized_elapsed_time = results[i - 1].elapsed_time *
      ( (double)num_nodes_visited_in_forward_traversal /
```

```
      (double)results[i - 1].num_parts_connected );
  }

  // report the benchmark results
  report_results( REVERSE_TRAVERSAL );
}
////////////////////////////////////////////////////////////////////////
```

2.3.9 params.hh

The source code for the file *params.hh* is listed below. This file defines several constant parameters which the OO1 benchmark implementation requires.

```
#ifndef __PARAMS_HH
#define __PARAMS_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 params.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 01 Jul 1993

 Revisions:
 15 Jul 93 -TJH- Added ObjectStore specific parameters.
*/
// definition of the number of connections for each part (3 for 001)
#define NUM_CONNECTIONS 3
// definition of the depth of the forward and reverse traversals (7 for 001)
#define TRAVERSAL_DEPTH 7
// definition of the locality of reference value (1 for 001 which means that
//  90% of the connections are randomly selected among the 1% of the parts
//  which are closest (in terms of part id), and the rest of the connections
//  are made to any randomly selected part.
#define LOCALITY_OF_REFERENCE 1
// definitions to create a 10 year range of dates
#define JAN_1_1980 315550800
#define JAN_1_1990 631170000
// 001 benchmark parameters
#define NUM_BENCHMARK_ITERATIONS 10
#define NUM_LOOKUP 1000
#define NUM_FORWARD_TRAVERSAL 1
#define NUM_REVERSE_TRAVERSAL 1
#define NUM_INSERT 100
// ObjectStore parameters
#define FETCH_SIZE_IN_BYTES 8192
#endif __PARAMS_HH
```

2.3.10 part.hh

The source code for the file *part.hh* is listed below. This file defines the *part* class.

```
#ifndef __PART_HH
#define __PART_HH
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
 ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

 part.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 18 Jun 1993

 Revisions:
 02 Jul 1993 -TJH- Added an index on the "id" attribute.
 21 Jul 1993 -TJH- Changed the type of "build" to long (from char) to
                   comply with the benchmark specification.
*/
extern os_database* db;
class connection;

class part {
public:
  persistent<db> os_Set<part*>* extent;
  int id indexable;
  char type[11];
  int x;
  int y;
  long build;
  os_Set<connection*> connected_to inverse_member from;
  os_Set<connection*> connected_from inverse_member to;
  part( int new_id, char *new_type, int new_x, int new_y,
    long new_build );
  ~part( );
  int forward_traversal( int current_level );
  int reverse_traversal( int current_level );
};
#endif __PART_HH
```

2.3.11 part.cc

The source code for the file *part.cc* is listed below. This file implements the methods for the *part* class.

```
/*

 #####   #####     #
#     # #     #   ##
#     # #     #  # #
#     # #     #    #    OBJECTSTORE
#     # #     #    #
#     # #     #    #
 #####   #####   #####

"Object Operations Benchmark" R.G.G. Cattell and J. Skeen
                              Sun Microsystems
```

```
ACM Transactions on Database Systems Vol. 17, No. 1, March 1992, Pages 1-31.

part.cc

Air Force Institute of Technology
Timothy J. Halloran
18 Jun 1993

Revisions:
01 Jul 1993 -TJH- The function "null_procedure_1()" was changed to
                  "null_procedure".
06 Jul 1993 -TJH- Changed all the debug output to use the "debug.hh" macros.
21 Jul 1993 -TJH- Changed the type of "build" to long (from char) to
                  comply with the benchmark specification.
29 Jul 1993 -TJH- Changed the constructor so that the "connected_to" and
                  "connected_from" attributes included the behavior
                  "os_collection::maintain_cursors."  This allows the
                  connection objects to delete themselves.
02 Aug 1993 -TJH- Moved the removal of connections from a part which is being
                  deleted from the "oo1::insert_measure( )" function to the
                  destructor of the part class.  The destructor function now
                  deletes all the connection objects which involve the
                  part being deleted.  This will keep the database
                  consistant.
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<debug.hh>
#include"connect.hh"
#include"part.hh"
#include"nullproc.hh"
#include"params.hh"
////////////////////////////////////////////////////////////////////////
part::part( int new_id, char *new_type, int new_x, int new_y,
  long new_build )
{
  DEBUG_INIT( "PART" , "part::part" );
  DEBUG_OUT( "entering", 1 );

  id = new_id;
  strcpy( type, new_type );
  x = new_x;
  y = new_y;
  build = new_build;
  connected_to.change_behavior( os_collection::maintain_cursors );
  connected_from.change_behavior( os_collection::maintain_cursors );
  extent->insert( this );
}
////////////////////////////////////////////////////////////////////////
part::~part( )
{
  DEBUG_INIT( "PART" , "part::~part" );
  DEBUG_OUT( "entering", 1 );

  DEBUG_OUT( sprintf( BUG, "deleting part %d", id ), 2 );

  DEBUG_OUT( sprintf( BUG, "deleting the %d \"connected_to\" connections",
    connected_to.cardinality( ) ), 2 );
  // delete all the connections from this part to other parts
  foreach ( connection *c, connected_to, os_cursor::safe ) {
    delete c;
  }
  DEBUG_OUT( sprintf( BUG, "deleting the %d \"connected_from\" connections",
```

```
    connected_from.cardinality( ) ), 2 );
  // delete all the connections to this part from other parts
  foreach ( c, connected_from, os_cursor::safe ) {
    delete c;
  }
  extent->remove( this );
}
////////////////////////////////////////////////////////////////////////
int part::forward_traversal( int current_level )
{
  DEBUG_INIT( "PART" , "part::forward_traversal" );
  DEBUG_OUT( "entering", 1 );

  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)", this->id, current_level ), 2 );
  // call the required null procedure
  null_procedure( this->x, this->y, this->type );
  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {
    foreach ( connection *c, this->connected_to ) {
      num_parts_connected += c->to->forward_traversal( current_level + 1 );
    }
  }
  return ( num_parts_connected + 1 );
}
////////////////////////////////////////////////////////////////////////
int part::reverse_traversal( int current_level )
{
  DEBUG_INIT( "PART" , "part::reverse_traversal" );
  DEBUG_OUT( "entering", 1 );

  DEBUG_OUT( sprintf( BUG, "part id %6d (level %d)", this->id, current_level ), 2 );
  // call the required null procedure
  null_procedure( this->x, this->y, this->type );
  int num_parts_connected = 0;
  if ( current_level < TRAVERSAL_DEPTH ) {
    foreach ( connection *c, this->connected_from ) {
      num_parts_connected += c->from->reverse_traversal( current_level + 1 );
    }
  }
  return ( num_parts_connected + 1 );
}
////////////////////////////////////////////////////////////////////////
```

3 Simulation Benchmark

This section lists the source code for our implementation of the simulation benchmark. An implementation was created for the ObjectStore object-oriented DBMS, and a non-persistent version was created. Only the ObjectStore implementation is listed due to the minor changes (removing the ObjectStore specific code) necessary to create the non-persistent version of the benchmark.

3.1 ObjectStore Implementation

This section lists the source code for the ObjectStore implementation of the simulation benchmark. The implementation was created on version 2.0.1 of ObjectStore using the C++ DML API. The executable *simbench* is the benchmark program, and the executable *create* sets up a database for use with the benchmark program.

3.1.1 Makefile

The *Makefile* for the ObjectStore SimBench source code is listed below.

```
include $(OS_ROOTDIR)/etc/ostore.mk
OS_COMPILATION_SCHEMA_DB_PATH=
  prowler:/usr3/databases/ostore_data/sim.comp_schema
OS_APPLICATION_SCHEMA_DB_PATH=
  prowler:/usr3/databases/ostore_data/sim.app_schema
#############################################################################
#
# If debug output is desired uncomment the "CCFLAGS" definition with
#  "-DDEBUG" in it.  Only one definition should be uncommented.
#
#CCFLAGS=-I. -DDEBUG -DXTFUNCPROTO -I/usr/questwin/include
  -I/usr/questwin/include/X11 -gx
CCFLAGS=-I. -DXTFUNCPROTO -I/usr/questwin/include -I/usr/questwin/include/X11
#
#############################################################################
LDFLAGS=-L/usr/questwin/lib/X11
XLIBS=-lXm -lXt -lX11
LDLIBS=-lm -loscol -los
SOURCES=aircraft.cc create.cc dice.cc expon.cc hex.c hexboard.cc hexplay.cc
  location.cc model.cc pmmlcg.c simbench.cc simulate.cc stopwtch.cc truck.cc
OBJECTS=aircraft.o create.o dice.o expon.o hex.o hexboard.o hexplay.o
  location.o model.o pmmlcg.o simbench.o simulate.o stopwtch.o truck.o
EXECUTABLES=simbench create

all: ${EXECUTABLES}

strange:
        OSCC -c pmmlcg.c
        cc -c -I/usr/questwin/include -I/usr/questwin/include/X11 hex.c

simbench: simbench.o simulate.o hexplay.o model.o aircraft.o truck.o
  hexboard.o location.o stopwtch.o dice.o expon.o pmmlcg.o hex.o
        ${LINK.cc} -o simbench \
          simbench.o simulate.o hexplay.o \
          model.o aircraft.o truck.o hexboard.o \
          location.o stopwtch.o dice.o expon.o pmmlcg.o hex.o \
          ${XLIBS} ${LDLIBS}

create: create.o
        ${LINK.cc} -o create create.o ${LDLIBS}

clean:
        rm -f ${EXECUTABLES} *.o
        osrm -f /usr3/databases/ostore_data/sim.comp_schema

depend:
        osmakedep .depend $(CCFLAGS) $(CPPFLAGS) -files $(SOURCES)

include .depend
```

3.1.2 aircraft.hh

The source code for the file *aircraft.hh* is listed below.

```
#ifndef ___AIRCRAFT_HH
#define ___AIRCRAFT_HH
/*

   #
  # #       #    #####    ####   #####     ##    ######   #####
```

```
 #   #      #    #    #  #    #  #    #   #  #   #          #
#     #     #    #    #  #       #    #  #    #  #####      #
#######     #    #####   #       #####   ######  #          #
#     #     #    #   #   #    #  #   #   #    #  #          #
#     #     #    #    #   ####   #    #  #    #  #          #

 aircraft.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include"player.hh"

class aircraft : public player {

  char name[10];
  char home_base[11];
  char location[10];
  char misc_sim_data[51];

public:

  aircraft( char *name, char *home_base, char *location, char *misc_sim_data );
  environment *clone( os_database *db, os_configuration *model_config );
  void execute( char *message, model *sim_model );
  char *query( char *data_item );
};
#endif ___AIRCRAFT_HH
```

3.1.3 aircraft.cc

The source code for the file *aircraft.cc* is listed below.

```
/*

   #
  # #       #    #####    ####   #####     ##    ######   #####
 #   #      #    #    #  #    #  #    #   #  #   #          #
#     #     #    #    #  #       #    #  #    #  #####      #
#######     #    #####   #       #####   ######  #          #
#     #     #    #   #   #    #  #   #   #    #  #          #
#     #     #    #    #   ####   #    #  #    #  #          #

 aircraft.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993

 Revisions:
 30 Sep 1993 -TJH- Removed all class inline functions due to compiler
                   limitations (CC compiler is not a "full" C++).
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"model.hh"
#include"aircraft.hh"
#include"dice.hh"
```

```
#include"expon.hh"
#include"location.hh"

#define STREAM 1
////////////////////////////////////////////////////////////////////////
aircraft::aircraft( char *name, char *home_base, char *location,
  char *misc_sim_data )
{
  DEBUG_INIT( "AIRCRAFT" , "aircraft::aircraft" );
  DEBUG_OUT( "entering", 1 );

  strncpy( aircraft::name, name, 9 );
  aircraft::name[9] = '\0';
  strncpy( aircraft::home_base, home_base, 10 );
  aircraft::home_base[10] = '\0';
  strncpy( aircraft::location, location, 9 );
  aircraft::location[9] = '\0';
  strncpy( aircraft::misc_sim_data, misc_sim_data, 50 );
  aircraft::misc_sim_data[50] = '\0';

  DEBUG_OUT( sprintf( BUG, "%s %s %s", name, home_base, location ), 2 );
}
////////////////////////////////////////////////////////////////////////
environment *aircraft::clone( os_database *db, os_configuration *model_config )
{
  DEBUG_INIT( "AIRCRAFT" , "aircraft::clone" );
  DEBUG_OUT( "entering", 1 );

  environment *c = new(db, model_config)
    aircraft( name, home_base, location, misc_sim_data );
  return c;
}
////////////////////////////////////////////////////////////////////////
void aircraft::execute( char *message, model *sim_model )
{
  DEBUG_INIT( "AIRCRAFT" , "aircraft::execute" );
  DEBUG_OUT( "entering", 1 );

  DEBUG_OUT( sprintf( BUG, "message is \"%s\"", message ), 2 );
  if ( !strcmp( message, "MOVE" ) ) {
    // execute the "MOVE" event
    int hexboard_width, hexboard_height;

    // determine the size of the hexmap
    for ( environment *e = sim_model->first_member ; e ; e = e->next ) {
      if ( !strcmp( e->query( "NAME" ), "HEXBOARD" ) ) {
        hexboard_width = atoi( e->query( "WIDTH" ) );
        hexboard_height = atoi( e->query( "HEIGHT" ) );
        break;
      }
    }

    // move the aircraft
    random_move( location, hexboard_width, hexboard_height );

    // schedule this aircraft to search the hex (exponential with mean 30.0)
    sim_model->schedule( this, name,
      sim_model->sim_time + (int)expon( 30.0 , STREAM ),
      "SEARCH" );
  }
  else if ( !strcmp( message, "SEARCH" ) ) {
    // execute the "SEARCH" event

    for ( environment *e = sim_model->first_member ; e ; e = e->next ) {
```

```
      if ( !strcmp( e->query( "LOCATION" ), location ) ) {
        if ( !strcmp( e->query( "CLASS" ), "TRUCK" ) ) {
          // located a truck in the same hex
          sim_model->add_logitem( name, e->query( "NAME" ), location,
            sim_model->sim_time );
        }
      }
    }

    // schedule the next move for this aircraft (exponential with mean 60.0)
    sim_model->schedule( this, name,
      sim_model->sim_time + (int)expon( 60.0 , STREAM ),
      "MOVE" );
  }
  else {
    fprintf( stderr, "WARNING[aircraft-%s] unknown message\n", name );
  }
}
////////////////////////////////////////////////////////////////////////
char *aircraft::query( char *data_item )
{
  DEBUG_INIT( "AIRCRAFT" , "aircraft::query" );
  DEBUG_OUT( "entering", 1 );

  if ( !strcmp( data_item, "CLASS" ) ) {
    return "AIRCRAFT";
  }
  else if ( !strcmp( data_item, "NAME" ) ) {
    return name;
  }
  else if ( !strcmp( data_item, "HOME_BASE" ) ) {
    return home_base;
  }
  else if ( !strcmp( data_item, "LOCATION" ) ) {
    return location;
  }
  return "ERROR";
}
////////////////////////////////////////////////////////////////////////
```

3.1.4 create.cc

The source code for the file *create.cc* is listed below.

```
/*

 #####
#     #  #####   ######    ##     #####  ######
#        #    #  #        #  #      #    #
#        #    #  #####   #    #     #    #####
#        #####   #       ######     #    #
#     #  #   #   #       #    #     #    #
 #####   #    #  ######  #    #     #    ######

 create.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 17 Aug 1993
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
```

```
#include<stdlib.h>
#include<debug.hh>
#include"model.hh"

#define TRUE 1
#define FALSE 0

os_database *db = 0;  // required to be here by ObjectStore
persistent<db> os_Set<model*> *model::extent = 0;

persistent<db> os_workspace *sim_ws = 0;
////////////////////////////////////////////////////////////////////////
main( int argc, char **argv)
{
  DEBUG_INIT( "MAIN" , "main" );
  DEBUG_OUT( "entering", 1 );

  // check the command line arguments
  if ( argc < 2) {
    fprintf( stderr, "create [database]\n" );
    return 1;  // exit the program
  }
  char *db_name = argv[1];

  // output a program title
  printf( "CREATE [OBJECTSTORE] Air Force Institute of Technology\n" );
  printf( " Creating ObjectStore database \"%s\".\n", db_name );

  objectstore::initialize( );
  os_collection::initialize( );

  // open the database
  db = os_database::create( db_name );

  do_transaction () {
    sim_ws = os_workspace::create_global( db );
    os_workspace::set_current( sim_ws );
  }

  do_transaction () {
    // create an extent (with an index on "name") for the model class
    model::extent = &os_Set<model*>::create( db, 0, (os_int32)100 );
  }

  db->close( );
}
////////////////////////////////////////////////////////////////////////
```

3.1.5 environ.hh

The source code for the file *environ.hh* is listed below.

```
#ifndef ___ENVIRONMENT_HH
#define ___ENVIRONMENT_HH
/*

#######
#       #    #  #    #     #    #####    ####   #    #
#       ##   #  #    #     #    #    #  #    #  ##   #
#####   # #  #  #    #     #    #    #  #    #  # #  #
#       #  # #  #    #     #    #####   #    #  #  # #
#       #   ##   #  #      #    #   #   #    #  #   ##
#######  #    #    ##      #    #    #   ####   #    #
```

```
 environ.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include<stdio.h>

class environment {

public:

  environment *next;

  virtual ~environment( )
  {
    // empty
  }

  virtual environment *clone( os_database *db, os_configuration *model_config )
  {
    environment *c = new(db, model_config) environment( );
    *c = *this;
    return c;
  }

  virtual char *query( char *data_item )
  {
    // this code should not be called
    fprintf( stderr, "WARNING[environment::query] base class function called\n" );
    return "ERROR";
  }
};
#endif ___ENVIRONMENT_HH
```

3.1.6 event.hh

The source code for the file *event.hh* is listed below.

```
#ifndef ___EVENT_HH
#define ___EVENT_HH
/*

#######
#       #    #  ######  #    #   #####
#       #    #  #       ##   #     #
#####   #    #  #####   # #  #     #
#       #    #  #       #  # #     #
#        #  #   #       #   ##     #
#######   ##    ######  #    #     #

 event.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include<stdio.h>
#include"player.hh"

class event {

public:
```

```
  char message[80];
  event *next;
  event *prev;
  int time;
  player *to_player;
  char to_player_name[10];


  event( player *to_player, char *to_player_name, int time, char *message )
  {
    event::to_player = to_player;
    strncpy( event::to_player_name, to_player_name, 9 );
    event::to_player_name[9] = '\0';
    event::time = time;
    strncpy( event::message, message, 79 );
    event::message[79] = '\0';
    event::next = event::prev = NULL;
  }
};
#endif ___EVENT_HH
```

3.1.7 hex.h

The source code for the file *hex.h* is listed below.

```
#ifndef _hex_h
#define _hex_h
/*

#     #
#     #  ######  #    #
#     #  #       #  #
#######  #####     ##
#     #  #         ##
#     #  #       #  #
#     #  ######  #    #

#     #
#  #  #     #    #####    ####   ######   #####
#  #  #     #    #    #  #    #  #          #
#  #  #     #    #    #  #       #####      #
#  #  #     #    #    #  #  ###  #          #
#  #  #     #    #    #  #    #  #          #
 ## ##      #    #####    ####   ######     #

 hex.h  (hex widget public header file)

 Air Force Wargaming Center
 Timothy J. Halloran
 September 1990
*/

/*
 Resources:
 ----------
 Name                Class             RepType         Default Value
 ----                -----             -------         -------------
 XtNbuttonCallback   Callback          Callback        NULL
 XtNdisplayHexLabels Boolean           Boolean         True
 XtNhexLabelFont     Font              XFontStruct*    XtDefaultFont
 XtNhexOutline       Color             Pixel           XtDefaultForeground
 XtNhexRadius        Width             Dimension       30
 XtNnumberHexX       Width             Dimension       10
 XtNnumberHexY       Height            Dimension       10
```

```
 XtNstackingDot      Color             Pixel          XtDefaultForeground


 Child Resources:
 ----------------
 Name                Class             RepType        Default Value
 ----                -----             -------        -------------
 XtNhexX             Position          int            1
 XtNhexY             Position          int            1
*/

/* special resource names not in <X11/StringDefs.h> */
#define XtNbuttonCallback               "buttonCallback"
#define XtNdisplayHexLabels             "displayHexLabels"
#define XtNhexLabelFont                 "hexLabelFont"
#define XtNhexOutline                   "hexOutline"
#define XtNhexRadius                    "hexRadius"
#define XtNnumberHexX                   "numberHexX"
#define XtNnumberHexY                   "numberHexY"
#define XtNstackingDot                  "stackingDot"

/* for children */
#define XtNhexX                         "hexX"
#define XtNhexY                         "hexY"

/* method declarations for the hex widget */
Widget HX_CreateHexWidget();
void   HX_WhichHex();

/* declare the class name */
#define hexClassName "hex"

/* declare specific hex widget class and instance datatypes */
typedef struct _HexClassRec* HexWidgetClass;
typedef struct _HexRec* HexWidget;

/* declare specific hex widget public structures */
#define MAXX 200 /* Maximum width of the hex board */
#define MAXY 200 /* Maximum height of the hex board */

typedef struct {
  int hexX;    /* X hex coordinate */
  int hexY;    /* Y hex coordinate */
} HXcoordinate;

/* declare the hex widget class constant */

extern WidgetClass hexWidgetClass;
#endif  _hex_h
```

3.1.8 hexP.h

The source code for the file *hexP.h* is listed below.

```
#ifndef _hexP_h
#define _hexP_h
/*

#     #
#     #  ######  #    #
#     #  #       #  #
#######  #####     ##
#     #  #         ##
#     #  #       #  #
```

```
#     #  ######  #    #

#     #
#  #  #     #    #####    ####   ######   #####
#  #  #     #    #    #  #    #  #          #
#  #  #     #    #    #  #       #####      #
#  #  #     #    #    #  #  ###  #          #
#  #  #     #    #    #  #    #  #          #
 ## ##      #    #####    ####   ######     #

 hexP.h  (hex widget private header file)

 Air Force Wargaming Center
 Timothy J. Halloran
 September 1990
*/
#include "hex.h"

/* include superclass private header file */
#include <X11/CoreP.h>
#include <X11/CompositeP.h>
#include <X11/ConstrainP.h>

#define MARGIN 3 /* margin around hex board (in pixels) */

/* functions in hex.c */
void             HX_ButtonHandler();
void             HX_CalculateHexLayout();
void             HX_ChangeManaged();
void             HX_CheckStacked();
void             HX_ConstraintInitialize();
Boolean          HX_ConstraintSetValues();
void             HX_Destroy();
void             HX_DrawSingleHex();
void             HX_Expose();
XtGeometryResult HX_GeometryManager();
void             HX_Initialize();
void             HX_Resize();

typedef struct {
    XPoint layout[8];                   /* 0 is center; 1-7 is hex outline  */
    int    stacked;                     /* Stackin flag.  True if more than */
                                        /*  one child, False if one or zero.*/
} HXhex;

typedef struct {
    int make_compiler_happy;
} HexClassPart;

typedef struct _HexClassRec {
    CoreClassPart       core_class;
    CompositeClassPart  composite_class;
    ConstraintClassPart constraint_class;
    HexClassPart        hex_class;
} HexClassRec;

extern HexClassRec hexClassRec;

typedef struct {

    /* resources */

    XtCallbackList button_callback;     /* The button callback list         */
    Boolean        display_hex_labels;  /* Are labels drawn on each hex?    */
```

```
                                        /*  (True or False)                */
    XFontStruct   *hex_label_font;      /* The font used for hex labels     */
    Pixel          hex_outline;         /* Color for drawing hex outlines   */
    Dimension      hex_radius;          /* Hex radius in pixels             */
    Dimension      number_hex_x;        /* Width of the hex board in hexes  */
    Dimension      number_hex_y;        /* Height of the hex board in hexes */
    Pixel          stacking_dot;        /* Color for the stacking dot       */

    /* private state */

    GC             gc_hex_outline;      /* Graphics context for drawing the */
                                        /*  hex outlines and labels.        */
    GC             gc_stacking_dot;     /* Graphics context for drawing the */
                                        /*  stacking dot                    */
    HXhex          hex_board[MAXX+1][MAXY+1]; /* The hex board              */
} HexPart;

typedef struct _HexRec {
    CorePart       core;
    CompositePart  composite;
    ConstraintPart constraint;
    HexPart        hex;
} HexRec;

typedef struct _HexConstraintsRec {
    HXcoordinate   position;
} HexConstraintsRec, *HexConstraints;
#endif  _hexP_h
```

3.1.9 hex.c

The source code for the file *hex.c* is listed below.

```
/*

#     #
#     #  ######  #    #
#     #  #        #  #
#######  #####     ##
#     #  #         ##
#     #  #        #  #
#     #  ######  #    #

#     #
#  #  #     #    #####    ####   ######   #####
#  #  #     #    #    #  #    #  #          #
#  #  #     #    #    #  #       #####      #
#  #  #     #    #    #  #  ###  #          #
#  #  #     #    #    #  #    #  #          #
 ## ##      #    #####    ####   ######     #

 hex.c

 Air Force Wargaming Center
 Timothy J. Halloran
 September 1990

 Revisions:
  02 Oct 1993 -TJH- Modified to minimize the huge amount of memory
                    used by the AFWC implementation.  This was done
                    to allow very large hex boards to be created.
                   (Air Force Institute of Technology)
  02 Oct 1993 -TJH- Created the "HX_Destroy" function to make sure that
                    memory is returned to the system when a hex is
```

```
                    destroyed (this was not being done, this widget
                    had LARGE memory leaks).
  02 Oct 1993 -TJH- The hex widget presently has a static maximum size.
                    It is defined by MAXX and MAXY in the "hex.h" file.
                    I added code to ensure the defined maximums are not
                    violated, but a dynamic solution would be MUCH better.
*/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <X11/IntrinsicP.h>
#include <X11/StringDefs.h>

#include "hexP.h"

static XtResource resources[] = {
#define offset(field) XtOffset(HexWidget, hex.field)
  /* {name, class, type, size, offset, default_type, default_addr}, */
  { XtNbuttonCallback, XtCCallback, XtRCallback, sizeof(XtCallbackList),
     offset(button_callback), XtRCallback,          NULL},
  { XtNdisplayHexLabels, XtCBoolean, XtRBoolean, sizeof(Boolean),
     offset(display_hex_labels), XtRString,         (caddr_t)"True"},
  { XtNhexLabelFont, XtCFont, XtRFontStruct, sizeof(XFontStruct *),
     offset(hex_label_font), XtRString,             "XtDefaultFont"},
  { XtNhexOutline, XtCColor, XtRPixel, sizeof(Pixel),
     offset(hex_outline), XtRString,                XtDefaultForeground},
  { XtNhexRadius, XtCWidth, XtRDimension, sizeof(Dimension),
     offset(hex_radius), XtRString,                 (caddr_t)"30"},
  { XtNnumberHexX, XtCWidth, XtRDimension, sizeof(Dimension),
     offset(number_hex_x), XtRString,               (caddr_t)"10"},
  { XtNnumberHexY, XtCHeight, XtRDimension, sizeof(Dimension),
     offset(number_hex_y), XtRString,               (caddr_t)"10"},
  { XtNstackingDot, XtCColor, XtRPixel, sizeof(Pixel),
     offset(stacking_dot), XtRString,               XtDefaultForeground},
#undef offset
};

static XtResource hexConstraintResources[] = {
  /* {name, class, type, size, offset, default_type, default_addr}, */
  { XtNhexX, XtCPosition, XtRInt, sizeof(int),
     XtOffset(HexConstraints, position.hexX), XtRString, (caddr_t)"1"},
  { XtNhexY, XtCPosition, XtRInt, sizeof(int),
     XtOffset(HexConstraints, position.hexY), XtRString, (caddr_t)"1"},
};

static XtActionsRec actions[] =
{
/*{name,            procedure}, */
  {"handle_button", HX_ButtonHandler},
};

static char translations[] =
" <BtnDown>: handle_button() \n\
";

HexClassRec hexClassRec = {
  { /* core fields */
    /* superclass               */      (WidgetClass) &constraintClassRec,
    /* class_name               */      hexClassName,
    /* widget_size              */      sizeof(HexRec),
    /* class_initialize         */      NULL,
    /* class_part_initialize    */      NULL,
    /* class_inited             */      FALSE,
    /* initialize               */      HX_Initialize,
```

```
    /* initialize_hook          */      NULL,
    /* realize                  */      XtInheritRealize,
    /* actions                  */      actions,
    /* num_actions              */      XtNumber(actions),
    /* resources                */      resources,
    /* num_resources            */      XtNumber(resources),
    /* xrm_class                */      NULLQUARK,
    /* compress_motion          */      TRUE,
    /* compress_exposure        */      TRUE,
    /* compress_enterleave      */      TRUE,
    /* visible_interest         */      FALSE,
    /* destroy                  */      HX_Destroy,
    /* resize                   */      HX_Resize,
    /* expose                   */      HX_Expose,
    /* set_values               */      NULL,
    /* set_values_hook          */      NULL,
    /* set_values_almost        */      XtInheritSetValuesAlmost,
    /* get_values_hook          */      NULL,
    /* accept_focus             */      NULL,
    /* version                  */      XtVersion,
    /* callback_private         */      NULL,
    /* tm_table                 */      translations,
    /* query_geometry           */      XtInheritQueryGeometry,
    /* display_accelerator      */      XtInheritDisplayAccelerator,
    /* extension                */      NULL,
  },
  {
    /* composite fields */
    /* geometry_manager         */      HX_GeometryManager,
    /* change_managed           */      NULL/*HX_ChangeManaged*/,
    /* insert_child             */      XtInheritInsertChild,
    /* delete_child             */      XtInheritDeleteChild,
    /* extension                */      NULL,
  },
  { /* constraint fields */
    /* subresources             */      hexConstraintResources,
    /* subresources_count       */      XtNumber(hexConstraintResources),
    /* constraint_size          */      sizeof(HexConstraintsRec),
    /* initialize               */      HX_ConstraintInitialize,
    /* destroy                  */      NULL,
    /* set_values               */      HX_ConstraintSetValues,
    /* extension                */      NULL,
  },
  { /* hex fields */
    /* make_compiler_happy      */      0,
  }
};

WidgetClass hexWidgetClass = (WidgetClass)&hexClassRec;

/*****************************************************************************/
void HX_ButtonHandler(hw, event, params, num_params)
HexWidget hw;
XEvent *event;
String *params;       /* unused */
Cardinal *num_params; /* unused */
{
  /* Call the XtNbuttonCallback callback list */
  XtCallCallbacks(hw, XtNbuttonCallback, (caddr_t)event);
}
/*****************************************************************************/
void HX_CalculateHexLayout(layout, quarter_hex_width, half_hex_height)
XPoint layout[];
int quarter_hex_width;
```

```
int half_hex_height;
{
  /* Calculate hex side points */
  layout[1].x = (layout[0].x-quarter_hex_width);
  layout[1].y = (layout[0].y-half_hex_height);
  layout[2].x = (layout[0].x+quarter_hex_width);
  layout[2].y = layout[1].y;
  layout[3].x = (layout[0].x+2*quarter_hex_width);
  layout[3].y = layout[0].y;
  layout[4].x = layout[2].x;
  layout[4].y = (layout[0].y+half_hex_height);
  layout[5].x = layout[1].x;
  layout[5].y = layout[4].y;
  layout[6].x = (layout[0].x-2*quarter_hex_width);
  layout[6].y = layout[0].y;
  layout[7] = layout[1];
}
/*****************************************************************************/
void HX_ChangeManaged(hw)
HexWidget hw;
{
  /* Do nothing */
}
/*****************************************************************************/
void HX_CheckStacked(hw, hexX, hexY, IsInit)
HexWidget hw;
int hexX, hexY;
int IsInit;
{
  HexConstraints child_const;
  int i, num_children;

  /* Determine how many children are in this hex */
  if (IsInit) /* If child is being initialized it does not show in list */
    num_children = 1;
  else
    num_children = 0;
  for (i=0; i<hw->composite.num_children; i++) {
    if (hw->composite.children[i]->core.managed){
      child_const = (HexConstraints)hw->composite.children[i]->core.constraints;
      if ((child_const->position.hexX == hexX)&&
          (child_const->position.hexY == hexY)) {
        num_children++;
      }
    }
  }

  if (num_children>1) {
    /* draw stacking dot */
    hw->hex.hex_board[hexX][hexY].stacked = True;
  }
  else {
    /* erase stacking dot */
    hw->hex.hex_board[hexX][hexY].stacked = False;
  }

  /* Redraw the hex */
  HX_DrawSingleHex(hw, hexX, hexY);
}
/*****************************************************************************/
void HX_ConstraintInitialize(request, new)
Widget request, new;
{
  HexConstraints hex_const = (HexConstraints) new->core.constraints;
```

```
  HexWidget hw = (HexWidget) new->core.parent;

  /* Check that the hexX and hexY are in range */
  if ((hex_const->position.hexX < 1)||
      (hex_const->position.hexX > hw->hex.number_hex_x)) {
    hex_const->position.hexX = 1;
  }
  if ((hex_const->position.hexY < 1)||
      (hex_const->position.hexY > hw->hex.number_hex_y)) {
    hex_const->position.hexY = 1;
  }

  /* Set the widget in the correct position */
  new->core.x = (Position)hw->hex.hex_board[hex_const->position.hexX]
                [hex_const->position.hexY].layout[0].x-(new->core.width/2);
  new->core.y = (Position)hw->hex.hex_board[hex_const->position.hexX]
                [hex_const->position.hexY].layout[0].y-(new->core.height/2);

  /* Check hex for stacked children */
  HX_CheckStacked(hw, hex_const->position.hexX, hex_const->position.hexY, True);
}
/*****************************************************************************/
Boolean HX_ConstraintSetValues(current, request, new)
Widget current, request, new;
{
  HexConstraints new_const = (HexConstraints) new->core.constraints;
  HexConstraints current_const = (HexConstraints) current->core.constraints;
  HexConstraints child_const;
  HexWidget hw = (HexWidget) new->core.parent;
  Position x, y;
  int i, num_children;

  /* Check that the latitude and longitude are in range */
  if ((new_const->position.hexX < 1)||
      (new_const->position.hexX > hw->hex.number_hex_x)) {
    new_const->position.hexX = current_const->position.hexX;
  }
  if ((new_const->position.hexY < 1)||
      (new_const->position.hexY > hw->hex.number_hex_y)) {
    new_const->position.hexY = current_const->position.hexY;
  }

  /* Set the widget in the correct position */
  x = (Position)(hw->hex.hex_board[new_const->position.hexX]
      [new_const->position.hexY].layout[0].x-(new->core.width/2));
  y = (Position)(hw->hex.hex_board[new_const->position.hexX]
      [new_const->position.hexY].layout[0].y-(new->core.height/2));
  XtMoveWidget(current, x, y);

  /* Check new and old hex for stacked children */
  HX_CheckStacked(hw,
                  new_const->position.hexX,
                  new_const->position.hexY,
                  False);
  HX_CheckStacked(hw,
                  current_const->position.hexX,
                  current_const->position.hexY,
                  False);

  return(False); /* redraw */
}
/*****************************************************************************/
Widget HX_CreateHexWidget(parent, name, arglist, argcount)
Widget parent;
```

```
String name;
ArgList arglist;
Cardinal argcount;
{
  return(XtCreateWidget(name, hexWidgetClass, parent, arglist, argcount));
}
/*****************************************************************************/
void HX_Destroy(hw)
HexWidget hw;
{
  /* release the GCs being used by the hex widget */
  XtReleaseGC(hw, hw->hex.gc_hex_outline);
  XtReleaseGC(hw, hw->hex.gc_stacking_dot);
}
/*****************************************************************************/
void HX_DrawSingleHex(hw, hexX, hexY)
HexWidget hw;
int hexX,hexY;
{
  int hex_label_len, hex_label_width;
  char hex_label[20];

  /* Only draw hex if widget is realized */
  if (XtIsRealized(hw)) {

    /* Draw the hex */

    /* 1. draw the stacking dot */
    if (hw->hex.hex_board[hexX][hexY].stacked) {
      XFillRectangle(XtDisplay(hw),
                     XtWindow(hw),
                     hw->hex.gc_stacking_dot,
                     (hw->hex.hex_board[hexX][hexY].layout[1].x + 3),
                     (hw->hex.hex_board[hexX][hexY].layout[1].y + 4),
                     4,
                     4);
    }

    /* 2. draw the hex outline */
    XDrawLines(XtDisplay(hw),
               XtWindow(hw),
               hw->hex.gc_hex_outline,
               &hw->hex.hex_board[hexX][hexY].layout[1],
               7, CoordModeOrigin);

    /* 3 draw the hex label */
    if (hw->hex.display_hex_labels) {
      sprintf(hex_label,"%d-%d",hexX,hexY);
      hex_label_len = strlen(hex_label);
      hex_label_width = XTextWidth(hw->hex.hex_label_font, hex_label,
                                   hex_label_len);
      XDrawString(XtDisplay(hw),
                  XtWindow(hw),
                  hw->hex.gc_hex_outline,
                  (hw->hex.hex_board[hexX][hexY].layout[0].x-
                   (int)(hex_label_width/2)),
                  (hw->hex.hex_board[hexX][hexY].layout[4].y-2),
                  hex_label,
                  hex_label_len);
    }
  }
}
/*****************************************************************************/
voi: HX_Expose(hw, event)
```

```
HexWidget hw;
XEvent *event;
{
  int loop_height, loop_width;

  /* Redraw the area */
  for (loop_width=1;loop_width<=hw->hex.number_hex_x;loop_width++) {
    for (loop_height=1;loop_height<=hw->hex.number_hex_y;loop_height++) {
      if ((hw->hex.hex_board[loop_width][loop_height].layout[3].x>=
           event->xexpose.x)&&
          (hw->hex.hex_board[loop_width][loop_height].layout[6].x<=
           (event->xexpose.x+event->xexpose.width))&&
          (hw->hex.hex_board[loop_width][loop_height].layout[4].y>=
           event->xexpose.y)&&
          (hw->hex.hex_board[loop_width][loop_height].layout[1].y<=
           (event->xexpose.y+event->xexpose.height)))
        HX_DrawSingleHex(hw, loop_width, loop_height);
    }
  }
}
/*****************************************************************************/
XtGeometryResult HX_GeometryManager(w, request, reply)
Widget w;
XtWidgetGeometry *request;
XtWidgetGeometry *reply;
{
  /* No position changes allowed! */
  if((request->request_mode & CWX && request->x != w->core.x)||
     (request->request_mode & CWY && request->y != w->core.y)) {
    return(XtGeometryNo);
  }
  return(XtGeometryYes);
}
/*****************************************************************************/
void HX_Initialize(request, new)
HexWidget request, new;
{
  int half_hex_height, quarter_hex_width;
  int loop_width, loop_height, i;
  XGCValues values;

  /* Check width and height of the hex board */
  if ((new->hex.number_hex_x<=0)||(new->hex.number_hex_y<=0))
    XtError("Hex board can't have zero height or width");
  if ((new->hex.number_hex_x>MAXX)||(new->hex.number_hex_y>MAXY))
    XtError("Hex board to large");

  /* Create outline GC */
  values.foreground = request->hex.hex_outline;
  values.font = request->hex.hex_label_font->fid;
  new->hex.gc_hex_outline = XtGetGC(new,
                                    (GCForeground|GCFont),
                                    &values);
  /* Create stacking dot GC */
  values.foreground = request->hex.stacking_dot;
  new->hex.gc_stacking_dot = XtGetGC(new,
                                     (GCForeground),
                                     &values);

  /* Set hex widget size */
  quarter_hex_width = (int)((sin((M_PI/180.0)*30.0)*
                             (double)new->hex.hex_radius)+0.5);
  half_hex_height = (int)(cos((M_PI/180.0)*30.0)*(double)new->hex.hex_radius);
```

```
  new->core.width = ((int)(new->hex.number_hex_x/2)*
                     (2*new->hex.hex_radius+2*quarter_hex_width))+
                     quarter_hex_width+
                    ((new->hex.number_hex_x%2)*3*quarter_hex_width)+(2*MARGIN);
  new->core.height= ((new->hex.number_hex_y*2*half_hex_height)
                     +half_hex_height)+(2*MARGIN);

  /* Create layout data for each hex */
  for (loop_width=1;loop_width<=new->hex.number_hex_x;loop_width++) {
    for (loop_height=1;loop_height<=new->hex.number_hex_y;loop_height++) {
      /* Calculate hex center point */
      new->hex.hex_board[loop_width][loop_height].layout[0].x =
        ((loop_width*new->hex.hex_radius)+(loop_width-1)*quarter_hex_width)
        +MARGIN;
      new->hex.hex_board[loop_width][loop_height].layout[0].y =
        (new->core.height-((loop_height*2*half_hex_height)-
        (loop_width%2)*half_hex_height))-MARGIN;
      HX_CalculateHexLayout(new->hex.hex_board[loop_width][loop_height].layout,
                            quarter_hex_width,
                            half_hex_height);
      /* Set stacking to false */
      new->hex.hex_board[loop_width][loop_height].stacked = False;
    }
  }
}
/*****************************************************************************/
void HX_Resize(hw)
HexWidget hw;
{
  /* Resize not supported */
}
/*****************************************************************************/
void HX_WhichHex(hw, x, y, hexX, hexY)
HexWidget hw;
int x, y, *hexX, *hexY;
{
  int loop_width, loop_height, cur_min_distance, distance;
  int deltaX, deltaY;

  cur_min_distance = 32767; /* a very large number */

  for (loop_width=1;loop_width<=hw->hex.number_hex_x;loop_width++) {
    for (loop_height=1;loop_height<=hw->hex.number_hex_y;loop_height++) {
      deltaX = (hw->hex.hex_board[loop_width][loop_height].layout[0].x - x);
      deltaY = (hw->hex.hex_board[loop_width][loop_height].layout[0].y - y);
      distance = (int) sqrt((double)(deltaX*deltaX+deltaY*deltaY));
      if (distance < cur_min_distance) {
        *hexX = loop_width; *hexY = loop_height;
        cur_min_distance = distance;
      }
    }
  }
}
/*****************************************************************************/
```

3.1.10 hexboard.hh

The source code for the file *hexboard.hh* is listed below.

```
#ifndef ___HEXBOARD_HH
#define ___HEXBOARD_HH
/*

#     #                ######
```

```
#     #  ######  #    # #     #   ####     ##    #####   #####
#     #  #       #  #  #     #  #    #   #  #   #    #  #    #
#######  #####     ##   ######  #    #  #    #  #    #  #    #
#     #  #         ##   #     #  #    #  ######  #####   #    #
#     #  #        #  #  #     #  #    #  #    #  #   #   #    #
#     #  ######  #    # ######    ####   #    #  #    #  #####

 hexboard.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include"environ.hh"

class hexboard : public environment {

  char name[10];
  int width;
  char width_str[10];
  int height;
  char height_str[10];

public:

  hexboard( char *name, int width, int height );
  environment *clone( os_database *db, os_configuration *model_config );
  char *query( char *data_item );
};
#endif ___HEXBOARD_HH
```

3.1.11 hexboard.cc

The source code for the file *hexboard.cc* is listed below.

```
/*

#     #                ######
#     #  ######  #    # #     #   ####     ##    #####   #####
#     #  #       #  #  #     #  #    #   #  #   #    #  #    #
#######  #####     ##   ######  #    #  #    #  #    #  #    #
#     #  #         ##   #     #  #    #  ######  #####   #    #
#     #  #        #  #  #     #  #    #  #    #  #   #   #    #
#     #  ######  #    # ######    ####   #    #  #    #  #####

 hexboard.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993

 Revisions:
 30 Sep 1993 -TJH- Removed all class inline functions due to compiler
                   limitations (CC compiler is not a "full" C++).
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"hexboard.hh"
//////////////////////////////////////////////////////////////////////////
```

```
hexboard::hexboard( char *name, int width, int height )
{
  DEBUG_INIT( "HEXBOARD" , "hexboard::hexboard" );
  DEBUG_OUT( "entering", 1 );

  strncpy( hexboard::name, name, 9 );
  hexboard::name[9] = '\0';
  hexboard::width = width;
  sprintf( width_str, "%d", width );
  hexboard::height = height;
  sprintf( height_str, "%d", height );

  DEBUG_OUT( sprintf( BUG, "%s %s %s", name, width_str, height_str ), 2 );
}
//////////////////////////////////////////////////////////////////////
environment *hexboard::clone( os_database *db, os_configuration *model_config )
{
  DEBUG_INIT( "HEXBOARD" , "hexboard::clone" );
  DEBUG_OUT( "entering", 1 );

  environment *c = new(db, model_config) hexboard( name, width, height );
  return c;
}
//////////////////////////////////////////////////////////////////////
char *hexboard::query( char *data_item )
{
  DEBUG_INIT( "HEXBOARD" , "hexboard::query" );
  DEBUG_OUT( "entering", 1 );

  if ( !strcmp( data_item, "CLASS" ) ) {
    return "HEXBOARD";
  }
  else if ( !strcmp( data_item, "NAME" ) ) {
    return "HEXBOARD";
  }
  else if ( !strcmp( data_item, "WIDTH" ) ) {
    return width_str;
  }
  else if ( !strcmp( data_item, "HEIGHT" ) ) {
    return height_str;
  }
  return "ERROR";
}
//////////////////////////////////////////////////////////////////////
```

3.1.12 hexplay.hh

The source code for the file *hexplay.hh* is listed below.

```
#ifndef ___HEXPLAY_HH
#define ___HEXPLAY_HH
/*

#     #                ######
#     #  ######  #    # #     #  #         ##     #   #
#     #  #       #  #  #     #  #        #  #     # #
#######  #####     ##   ######   #       #    #     #
#     #  #         ##   #        #       ######     #
#     #  #        #  #  #        #       #    #     #
#     #  ######  #    # #        ######  #    #     #

 hexplay.hh

 Air Force Institute of Technology
```

```
 Timothy J. Halloran
 30 Sep 1993
*/
void add_player_to_hexboard( char *player_class, char *player_name,
  int pos_x, int pos_y );
#endif ___HEXPLAY_HH
```

3.1.13 hexplay.cc

The source code for the file *hexplay.cc* is listed below.

```
/*

#     #                ######
#     #  ######  #    # #     #  #         ##     #   #
#     #  #       #  #  #     #  #        #  #     # #
#######  #####     ##   ######   #       #    #     #
#     #  #         ##   #        #       ######     #
#     #  #        #  #  #        #       #    #     #
#     #  ######  #    # #        ###### #    #     #

 hexplay.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 30 Sep 1993
*/
#include<stdio.h>
#include<stdlib.h>
#include<Xm/PushB.h>
#include<hex.h>
#include"hexplay.hh"

void lower_symbol_cb( Widget w, XtPointer closure, XtPointer call_data );

extern Widget map;
////////////////////////////////////////////////////////////////////////
void add_player_to_hexboard( char *player_class, char *player_name,
  int pos_x, int pos_y )
{
  XmString label = XmStringCreateSimple( player_name );

  Widget symbol = XtVaCreateManagedWidget( player_class,
    xmPushButtonWidgetClass, map,
    XtNhexX, pos_x,
    XtNhexY, pos_y,
    XmNlabelString, label,
    XmNhighlightThickness, 0,
    NULL );
  XmStringFree( label );
  XtAddCallback( symbol, XmNactivateCallback,
    lower_symbol_cb, NULL );
}
////////////////////////////////////////////////////////////////////////
void lower_symbol_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  // lower the symbol so that another can be seen if there are more than one
  XLowerWindow( XtDisplay( w ), XtWindow( w ) );
}
////////////////////////////////////////////////////////////////////////
```

3.1.14 location.hh

The source code for the file *location.hh* is listed below.

```
#ifndef __LOCATION_HH
#define __LOCATION_HH
/*

#
#         ####    ####     ##     #####     #     ####   #    #
#        #    #  #    #   #  #      #       #    #    #  ##   #
#        #    #  #       #    #     #       #    #    #  # #  #
#        #    #  #       ######     #       #    #    #  #  # #
#        #    #  #    #  #    #     #       #    #    #  #   ##
#######   ####    ####   #    #     #       #     ####   #    #

 location.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 29 Sep 1993
*/
void decode_location( char *location, int& pos_x, int& pos_y );
void encode_location( char *location, int pos_x, int pos_y );
void random_move( char *location, int hexboard_width, int hexboard_height );
#endif __LOCATION_HH
```

3.1.15 location.cc

The source code for the file *location.cc* is listed below.

```
/*

#
#         ####    ####     ##     #####     #     ####   #    #
#        #    #  #    #   #  #      #       #    #    #  ##   #
#        #    #  #       #    #     #       #    #    #  # #  #
#        #    #  #       ######     #       #    #    #  #  # #
#        #    #  #    #  #    #     #       #    #    #  #   ##
#######   ####    ####   #    #     #       #     ####   #    #

 location.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 29 Sep 1993
*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"location.hh"
#include"dice.hh"

#define STREAM 1
////////////////////////////////////////////////////////////////////////
void decode_location( char *location, int& pos_x, int& pos_y )
{
  DEBUG_INIT( "LOCATION" , "decode_location" );
  DEBUG_OUT( "entering", 1 );
  char pos_x_str[10];
  char *pos_y_str;

  strcpy( pos_x_str, location );
```

```
  pos_x_str[4] = '\0';
  pos_y_str = &pos_x_str[5];
  pos_x = atoi( pos_x_str );
  pos_y = atoi( pos_y_str );

  DEBUG_OUT( sprintf( BUG, "pos_x %d pos_y %d", pos_x, pos_y ), 2 );
}
////////////////////////////////////////////////////////////////////////
void encode_location( char *location, int pos_x, int pos_y )
{
  DEBUG_INIT( "LOCATION" , "encode_location" );
  DEBUG_OUT( "entering", 1 );

  sprintf( location, "%04d-%04d", pos_x, pos_y );

  DEBUG_OUT( sprintf( BUG, "%s", location ), 2 );
}
////////////////////////////////////////////////////////////////////////
void random_move( char *location, int hexboard_width, int hexboard_height )
{
  DEBUG_INIT( "LOCATION" , "random_move" );
  DEBUG_OUT( "entering", 1 );
  int pos_x, pos_y;

  DEBUG_OUT( sprintf( BUG, "old location %s", location ), 2 );
  decode_location( location, pos_x, pos_y );

  // make a random move (six possible directions)
  switch ( roll( 1, 6, STREAM ) ) {
  case 1:
    pos_y++; // move up
    break;
  case 2:
    pos_x++; // move right
    break;
  case 3:
    if ( pos_x % 2 )
      pos_y--;  // move down for an odd hex location
    else
      pos_y++;  // move up for an even hex location
    pos_x++; // move right
    break;
  case 4:
    pos_y--; // move down
    break;
  case 5:
    pos_x--; // move left
    break;
  case 6:
    if ( pos_x % 2 )
      pos_y--;  // move down for an odd hex location
    else
      pos_y++;  // move up for an even hex location
    pos_x--; // move left
    break;
  }

  // make sure the player stays on the hexboard
  if ( pos_x > hexboard_width ) pos_x = 1;
  if ( pos_x < 1 ) pos_x = hexboard_width;
  if ( pos_y > hexboard_height ) pos_y = 1;
  if ( pos_y < 1 ) pos_y = hexboard_height;

  encode_location( location, pos_x, pos_y );
```

```
  DEBUG_OUT( sprintf( BUG, "new location %s", location ), 2 );
}
////////////////////////////////////////////////////////////////////////
```

3.1.16 logitem.hh

The source code for the file *logitem.hh* is listed below.

```
#ifndef ___LOGITEM_HH
#define ___LOGITEM_HH
/*

#                        ###
#         ####    ####     #      #####  ######  #    #
#        #    #  #    #    #        #    #       ##  ##
#        #    #  #         #        #    #####   # ## #
#        #    #  #  ###    #        #    #       #    #
#        #    #  #    #    #        #    #       #    #
#######   ####    ####    ###       #    ######  #    #

 logitem.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
class logitem {

public:

  char search_aircraft[10];
  char truck_found[10];
  char location_found[10];
  int time;
  logitem *next;

  logitem( char *search_aircraft, char *truck_found, char *location_found,
    int time )
  {
    strncpy( logitem::search_aircraft, search_aircraft, 9 );
    logitem::search_aircraft[9] = '\0';
    strncpy( logitem::truck_found, truck_found, 9 );
    logitem::truck_found[9] = '\0';
    strncpy( logitem::location_found, location_found, 9 );
    logitem::location_found[9] = '\0';
    logitem::time = time;
  }

  void output( FILE *outfile )
  {
    fprintf( outfile, "%10s | %10s | %10s | %6d sec\n", search_aircraft,
      truck_found, location_found, time );
  }
};
#endif ___LOGITEM_HH
```

3.1.17 model.hh

The source code for the file *model.hh* is listed below.

```
#ifndef ___MODEL_HH
#define ___MODEL_HH
/*
```

```
#     #
##   ##   ####   #####   ######  #
# # # #  #    #  #    #  #       #
#  #  #  #    #  #    #  #####   #
#     #  #    #  #    #  #       #
#     #  #    #  #    #  #       #
#     #   ####   #####   ######  ######

 model.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include"environ.hh"
#include"event.hh"
#include"player.hh"
#include"logitem.hh"

extern os_database *db;

class model {

public:

  char name[13];
  int sim_time;
  os_Set<environment*> members;
  os_Set<event*> events;
  event *first_event;
  event *last_event;
  environment *first_member;
  environment *last_member;
  logitem *first_logitem;
  logitem *last_logitem;
  os_configuration *model_config;
  persistent<db> os_Set<model*> *extent;

  model( char *name, os_configuration *model_config );
  model( model *old_model, char *name, os_configuration *model_config );
  ~model( );
  void clear( );
  void run_until_sim_goal_time( int goal_sim_time );
  void add_environment_member( environment* new_member );
  void add_logitem( char *search_aircraft, char *truck_found,
    char *location_found, int time );
  void schedule( player *to_player, char* to_player_name,
    int at_time, char *message );
};
#endif ___MODEL_HH
```

3.1.18 model.cc

The source code for the file *model.cc* is listed below.

```
/*

#     #
##   ##   ####   #####   ######  #
# # # #  #    #  #    #  #       #
#  #  #  #    #  #    #  #####   #
#     #  #    #  #    #  #       #
#     #  #    #  #    #  #       #
```

```
#     #   ####   #####   ######  ######

 model.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"environ.hh"
#include"event.hh"
#include"player.hh"
#include"model.hh"
#include"logitem.hh"

#define TRUE  1
#define FALSE 0

extern os_database *db;
////////////////////////////////////////////////////////////////////////
model::model( char *name, os_configuration *model_config )
{
  DEBUG_INIT( "MODEL" , "model::model" );
  DEBUG_OUT( "entering", 1 );

  // create the new model (it starts off empty)
  strncpy( model::name, name, 12 );
  model::name[12] = '\0';
  model::model_config = model_config;
  sim_time = 0;
  first_event = last_event = NULL;
  first_member = last_member = NULL;
  first_logitem = last_logitem = NULL;

  extent->insert( this );
}
////////////////////////////////////////////////////////////////////////
model::model( model *old_model, char *name, os_configuration *model_config )
{
  DEBUG_INIT( "MODEL" , "model::model(copy)" );
  DEBUG_OUT( "entering", 1 );

  // create the new model
  strncpy( model::name, name, 12 );
  model::name[12] = '\0';
  model::model_config = model_config;
  first_event = last_event = NULL;
  first_member = last_member = NULL;
  first_logitem = last_logitem = NULL;

  // copy the simulation time
  sim_time = old_model->sim_time;

  // copy all the evironment members
  for ( environment *m = old_model->first_member ; m ; m = m->next ) {
    add_environment_member( m->clone( db, model_config ) );
  }
```

```
  // copy all the events
  for ( event *e = old_model->first_event ; e ; e = e->next ) {
    // find the handle to the new player in this copy of the model
    int scheduled_event = FALSE;
    for ( m = first_member ; m ; m = m->next ) {
      if ( !strcmp( m->query( "NAME"), e->to_player_name ) ) {
        player *p = (player*)m;
        schedule( p, e->to_player_name, e->time, e->message );
        scheduled_event = TRUE;
      }
    }
    if ( !scheduled_event ) {
      fprintf( stderr, "ERROR[model::model(copy)] unable to copy an event\n" );
      exit( 1 );
    }
  }

  // copy all the logitems
  for ( logitem *l = old_model->first_logitem ; l ; l = l->next ) {
    add_logitem( l->search_aircraft, l->truck_found, l->location_found, l->time );
  }

  extent->insert( this );
}
//////////////////////////////////////////////////////////////////////////
model::~model( )
{
  DEBUG_INIT( "MODEL" , "model::~model" );
  DEBUG_OUT( "entering", 1 );

  clear( );

  extent->remove( this );
}
//////////////////////////////////////////////////////////////////////////
void model::clear( )
{
  DEBUG_INIT( "MODEL" , "model::clear" );
  DEBUG_OUT( "entering", 1 );
  environment *temp_m;
  event *temp_e;
  logitem *temp_l;

  // delete all the environment members
  for ( environment *m = first_member ; m ; ) {
    temp_m = m->next;
    delete m;
    m = temp_m;
  }
  members.clear( );
  first_member = last_member = NULL;

  // delete all the events
  for ( event *e = first_event ; e ; ) {
    temp_e = e->next;
    delete e;
    e = temp_e;
  }
  events.clear( );
  first_event = last_event = NULL;

  // delete all the logitems
  for ( logitem *l = first_logitem ; l ; ) {
    temp_l = l->next;
```

```
    delete l;
    l = temp_l;
  }
  first_logitem = last_logitem = NULL;

  // reset the simulation time
  sim_time = 0;
}
////////////////////////////////////////////////////////////////////
void model::run_until_sim_goal_time( int goal_sim_time )
{
  DEBUG_INIT( "MODEL" , "model::run_until_sim_goal_time" );
  DEBUG_OUT( "entering", 1 );
  event *executing_event;

  // do nothing if the model is asked to run backwards
  if ( sim_time >= goal_sim_time ) return;

  // simulate
  while ( sim_time < goal_sim_time ) {
    if ( first_event == NULL ) {
      // there are no events left so update time and exit
      sim_time = goal_sim_time;
    }
    else if ( first_event->time >= goal_sim_time ) {
      // the next event does not occur before the goal time
      sim_time = goal_sim_time;
    }
    else {
      // execute a single event
      executing_event = first_event;
      first_event = first_event->next;
      if ( first_event == NULL ) {
        // there are no more events
        last_event = NULL;
      }
      else {
        // there is at least one remaining event in the event list
        first_event->prev = NULL;
      }
      sim_time = executing_event->time;
      // execute the event
      DEBUG_OUT( sprintf( BUG, "%s executing \"%s\"",
        executing_event->to_player_name, executing_event->message ), 2 );
      executing_event->to_player->execute( executing_event->message, this );
      // destroy the event (no persistant object leaks please)
      events.remove( executing_event );
      delete executing_event;
    }
  }
}
////////////////////////////////////////////////////////////////////
void model::add_environment_member( environment* new_member )
{
  DEBUG_INIT( "MODEL" , "model::add_environment_member" );
  DEBUG_OUT( "entering", 1 );

  // insert into the ObjectStore database set of members
  members.insert( new_member );

  // insert the new environment member on the member list
  if ( first_member == NULL ) {
    // insert the member as the only member on the member list
    first_member = last_member = new_member;
```

```
    new_member->next = NULL;
  }
  else {
    // insert the new member at the end of the member list
    last_member->next = new_member;
    last_member = new_member;
    new_member->next = NULL;
  }
}
////////////////////////////////////////////////////////////////////////
void model::add_logitem( char *search_aircraft, char *truck_found,
  char *location_found, int time )
{
  DEBUG_INIT( "MODEL" , "model::add_logitem" );
  DEBUG_OUT( "entering", 1 );

  // create a new logitem
  logitem *new_logitem = new (db,model_config) logitem( search_aircraft,
    truck_found, location_found, time );

  // insert the new logitem on the logitem list
  if ( first_logitem == NULL ) {
    // insert the logitem as the only logitem on the list
    first_logitem = last_logitem = new_logitem;
    new_logitem->next = NULL;
  }
  else {
    // insert the new logitem at the end of the list
    last_logitem->next = new_logitem;
    last_logitem = new_logitem;
    new_logitem->next = NULL;
  }
}
////////////////////////////////////////////////////////////////////////
void model::schedule( player *to_player, char* to_player_name,
  int at_time, char *message )
{
  DEBUG_INIT( "MODEL" , "model::schedule" );
  DEBUG_OUT( "entering", 1 );

  // create a new event
  event *new_event = new(db,model_config) event( to_player, to_player_name,
    at_time, message );

  // insert into the ObjectStore database set of events
  events.insert( new_event );

  // insert the new event on the event list
  if ( first_event == NULL ) {
    // insert the event as the only event on the event list
    first_event = last_event = new_event;
    new_event->next = new_event->prev = NULL;
  }
  else {
    // insert the event in the event list in time order
    for ( event *e = last_event ; e ; e = e->prev ) {
      if ( e->time <= new_event->time ) {
        // insert the event after "e"
        new_event->next = e->next;
        new_event->prev = e;
        e->next = new_event;
        if ( e == last_event ) {
          // this item is now the last item
          last_event = new_event;
```

```
        }
        else {
          new_event->next->prev = new_event;
        }
        break;
      }
      else if ( e == first_event ) {
        // insert the event at the very beginning of the event list
        new_event->prev = e->prev;
        new_event->next = e;
        e->prev = new_event;
        first_event = new_event;
        break;
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////
```

3.1.19 player.hh

The source code for the file *player.hh* is listed below.

```
#ifndef ___PLAYER_HH
#define ___PLAYER_HH
/*

######
#     #  #         ##     #   #  ######  #####
#     #  #        #  #     # #   #       #    #
######   #       #    #     #    #####   #    #
#        #       ######     #    #       #####
#        #       #    #     #    #       #   #
#        ######  #    #     #    ######  #    #

 player.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include<stdio.h>
#include"environ.hh"

class model;

class player : public environment {

public:

  virtual ~player( )
  {
    // empty
  }

  virtual void execute( char *message, model* sim_model )
  {
    // this code should not be called
    fprintf( stderr, "WARNING[player::execute] base class function called\n" );
  }
};
#endif ___PLAYER_HH
```

3.1.20 simbench.cc

The source code for the file *simbench.cc* is listed below.

```
/*

   #    #######   ###   #######
  # #   #          #       #
 #   #  #          #       #
#     # #####      #       #
####### #          #       #
#     # #          #       #
#     # #         ###      #


 #####                  ######
#     #    #    #    # #     #  ######  #    #   ####   #    #
#          #    ##  ## #     #  #       ##   #  #    #  #    #
 #####     #    # ## # ######   #####   # #  #  #       ######
      #    #    #    # #     #  #       #  # #  #       #    #
#     #    #    #    # #     #  #       #   ##  #    #  #    #
 #####     #    #    # ######   ######  #    #   ####   #    #

 simbench.cc

 This module contains all the user interface code.  The user interface
  uses the OSF Motif look and feel.  Very little in this module should
  need to be changed to port the "simbench" program to a new
  object-oriented DBMS.

 Air Force Institute of Technology
 Timothy J. Halloran
 15 Aug 1993

 Revisions:
 24 Oct 1993 -TJH- Fixed a bug in the run until timing, the timer was
                   waiting until the user pressed the OK button.
*/
#include<stdio.h>
#include<stdlib.h>
#include<sys/time.h>
#include<sys/resource.h>
#include<stdarg.h>
#include<Xm/DialogS.h>
#include<Xm/Form.h>
#include<Xm/Frame.h>
#include<Xm/Label.h>
#include<Xm/MainW.h>
#include<Xm/MessageB.h>
#include<Xm/PanedW.h>
#include<Xm/PushB.h>
#include<Xm/RowColumn.h>
#include<Xm/ScrolledW.h>
#include<Xm/SelectioB.h>
#include<Xm/Text.h>
#include<hex.h>
#include"simulate.hh"
#include"stopwtch.hh"
extern "C" int getpagesize( );  // not defined in a header file

// declare functions used in this module
////////// Utility and Setup Routines //////////
void create_map_dialog( );
void create_running_dialog( );
double get_clock_time( );
```

```
void set_starting_menu_state( );
void setup_main_window( );
void update_map_dialog( );
////////// User Information Routines //////////
void log_print( char *fmt, ... );
void msg_print( char *fmt, ... );
void update_model_msg( int model_exists );
////////// Xt Callback Support Routines //////////
char *get_about_message( );
Widget get_top_shell( Widget w );
////////// Xt Callback Routines //////////
void done_running_cb( Widget w, XtPointer closure, XtPointer call_data );
void menu_configure_cb( Widget w, XtPointer closure, XtPointer call_data );
void menu_execute_cb( Widget w, XtPointer closure, XtPointer call_data );
void menu_file_cb( Widget w, XtPointer closure, XtPointer call_data );
void menu_help_cb( Widget w, XtPointer closure, XtPointer call_data );
void menu_post_cb( Widget w, XtPointer closure, XtPointer call_data );
void take_down_cb( Widget w, XtPointer closure, XtPointer call_data );
void take_down_named_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_connect_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_create_scenario_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_new_model_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_open_model_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_run_until_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_save_as_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_set_time_ratio_cb( Widget w, XtPointer closure, XtPointer call_data );
void try_set_time_slice_cb( Widget w, XtPointer closure, XtPointer call_data );
void update_map_cb( Widget w, XtPointer closure, XtPointer call_data );
////////// Xt Work Procedures //////////
Boolean run_until_work_proc( XtPointer client_data );
Boolean run_work_proc( XtPointer client_data );
////////// Generic Ask User Routines //////////
int generic_ask_user( unsigned char dialog_type, char *question,
  char *answer_yes, char *answer_no, int default_answer );
//////////

#define HEX_RADIUS 40
#define TITLE_MSG            "AFIT Simulation Benchmark"
#define VERSION_MSG          "ObjectStore Version"
#define CONNECT_DIALOG_MSG   "Enter ObjectStore database filename:"
#define SLICE_DIALOG_MSG     "Enter desired time slice:"
#define RATIO_DIALOG_MSG     "Enter desired time ratio:"
#define RUN_UNTIL_DIALOG_MSG "Enter desired goal time for run:"

String fallbacks[ ] = {
  "SimBench*background: #ffc633",
  "SimBench*foreground: black",
  "SimBench*fontList:   -*-helvetica-medium-r-*-*-18-*-*-*-*-*-*-*-*",
  "SimBench*hexboard.background:   forestgreen",
  "SimBench*hexboard.hexLabelFont: fixed",
  "SimBench*hexboard.hexOutline:   white",
  "SimBench*hexboard*fontList:     fixed",
  "SimBench*hexboard*AIRCRAFT*background: grey",
  "SimBench*hexboard*TRUCK*background:    goldenrod",
  "SimBench*log*fontList: fixed",
  NULL,
};

char *scenario_prompts[] = {
  "Number aircraft:",
  "Number trucks:",
  "Hexboard size:",
};
```

```
// used by generic_ask_user
#define YES 1
#define NO  2

// global and module variables
static XtAppContext app;
static Widget toplevel;
static Widget message;
static Widget log;
static Widget running_dialog;
static Widget map_dialog = NULL;
static Widget map_control;
       Widget map = NULL;  // visible to "hexplay.cc"
static int run_until_goal_time;
static stopwatch *global_timer;
static double next_clock_time_to_run;
static double last_run_clock_time;
////////////////////////////////////////////////////////////////////////
void main( int argc, char **argv )
{
  toplevel = XtVaAppInitialize( &app, "SimBench", NULL, 0,
    (Cardinal *)&argc, argv, fallbacks, NULL );
  XtVaSetValues( toplevel,
    XtNtitle, "AFIT SimBench - ObjectStore Version",
    NULL );

  setup_main_window( );
  set_starting_menu_state( );  // Not all menu choices are sensitive
  update_model_msg( FALSE );

  XtRealizeWidget( toplevel );

  create_running_dialog( );

  XtAppMainLoop( app );
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////
////  Utility and Setup Routines
////
////////////////////////////////////////////////////////////////////////
void create_map_dialog( )
{
  map_dialog = XtVaCreatePopupShell( "map",
    xmDialogShellWidgetClass, toplevel,
    XtNtitle, "View Map",
    XmNmappedWhenManaged, False,
    NULL );
  Widget pane = XtVaCreateWidget( "pane", xmPanedWindowWidgetClass,
    map_dialog,
    XmNsashWidth, 1,
    XmNsashHeight, 1,
    NULL );

  // set up the control area of the dialog
  map_control = XtVaCreateManagedWidget( "control",
    xmScrolledWindowWidgetClass,
    pane,
    XmNheight, 400,
    XmNscrollingPolicy, XmAUTOMATIC,
    NULL );
  // set up the action area of the dialog
```

```
  Widget action = XtVaCreateWidget( "action", xmFormWidgetClass, pane,
    XmNfractionBase, 7,
    NULL );
  Widget ok = XtVaCreateManagedWidget( "OK",
    xmPushButtonWidgetClass, action,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_POSITION,
    XmNleftPosition, 1,
    XmNrightAttachment, XmATTACH_POSITION,
    XmNrightPosition, 2,
    XmNshowAsDefault, True,
    XmNdefaultButtonShadowThickness, 1,
    NULL );
  XtAddCallback( ok, XmNactivateCallback,
    take_down_named_cb, "map" );
  Widget update = XtVaCreateManagedWidget( "Update",
    xmPushButtonWidgetClass, action,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_POSITION,
    XmNleftPosition, 3,
    XmNrightAttachment, XmATTACH_POSITION,
    XmNrightPosition, 4,
    XmNdefaultButtonShadowThickness, 1,
    NULL );
  XtAddCallback( update, XmNactivateCallback,
    update_map_cb, "map" );
  Widget help = XtVaCreateManagedWidget( "Help",
    xmPushButtonWidgetClass, action,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_POSITION,
    XmNleftPosition, 5,
    XmNrightAttachment, XmATTACH_POSITION,
    XmNrightPosition, 6,
    XmNdefaultButtonShadowThickness, 1,
    NULL );
  XtSetSensitive( help, False );
  XtManageChild( action );
  XtManageChild( pane );

  // make sure that the action area can be always be seen
  XtWidgetGeometry size;
  size.request_mode = CWHeight;
  XtQueryGeometry( action, NULL, &size );
  XtVaSetValues( action,
    XmNpaneMaximum, size.height,
    XmNpaneMinimum, size.height,
    NULL );
}
////////////////////////////////////////////////////////////////////////
void create_running_dialog( )
{
  Arg args[3];
  XtSetArg( args[0], XmNdialogStyle, XmDIALOG_FULL_APPLICATION_MODAL );
  XtSetArg( args[1], XmNautoUnmanage, False );
  XtSetArg( args[2], XmNnoResize, True );
  running_dialog = XmCreateWorkingDialog( toplevel, "running", args, 3 );
  XtUnmanageChild( XmMessageBoxGetChild( running_dialog, XmDIALOG_OK_BUTTON ) );
  XtUnmanageChild( XmMessageBoxGetChild( running_dialog, XmDIALOG_HELP_BUTTON ) );
  XtVaSetValues( XtParent( running_dialog ),
    XtNtitle, "Running Simulation",
    NULL );
```

```
}
////////////////////////////////////////////////////////////////////
double get_clock_time( )
{
  struct timeval t;

  // read the current clock time and return it
  if ( gettimeofday( &t, (struct timezone*)0 ) ) {
    fprintf( stderr, "ERROR[get_clock_time] unable to read time\n" );
    exit( 1 );
  }
  return ( (double)t.tv_sec + (double)t.tv_usec / 1000000.0 );
}
////////////////////////////////////////////////////////////////////
void set_starting_menu_state( )
{
  Widget a_menu;

  // Set parts of the the "File" menu insensitive
  if ( a_menu = XtNameToWidget( toplevel,  // Disconnect
    "main_window.menu_bar.popup_file_menu.file_menu.button_1" ) )
    XtSetSensitive( a_menu, False );
  if ( a_menu = XtNameToWidget( toplevel,  // New Model...
    "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
    XtSetSensitive( a_menu, False );
  if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
    "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
    XtSetSensitive( a_menu, False );
  if ( a_menu = XtNameToWidget( toplevel,  // Close
    "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
    XtSetSensitive( a_menu, False );
  if ( a_menu = XtNameToWidget( toplevel,  // Save As...
    "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
    XtSetSensitive( a_menu, False );

  // Set the entire "Configure" menu insensitive
  if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
    XtSetSensitive( a_menu, False );

  // Set the entire "Execute" menu insensitive
  if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
    XtSetSensitive( a_menu, False );

  // Set the entire "Post Process" menu insensitive
  if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
    XtSetSensitive( a_menu, False );
}
////////////////////////////////////////////////////////////////////
void setup_main_window( )
{
  // global variables: Widget toplevel
  Widget main_window = XtVaCreateManagedWidget( "main_window",
    xmMainWindowWidgetClass, toplevel,
    NULL );

  XmString file = XmStringCreateSimple( "File" );
  XmString configure = XmStringCreateSimple( "Configure" );
  XmString execute = XmStringCreateSimple( "Execute" );
  XmString post = XmStringCreateSimple( "Post Process" );
  XmString help = XmStringCreateSimple( "Help" );
  Widget menu_bar = XmVaCreateSimpleMenuBar( main_window, "menu_bar",
    XmVaCASCADEBUTTON, file, 'F',
    XmVaCASCADEBUTTON, configure, 'C',
    XmVaCASCADEBUTTON, execute, 'E',
```

```
  XmVaCASCADEBUTTON, post, 'P',
  XmVaCASCADEBUTTON, help, 'H',
  NULL );
XmStringFree( file );
XmStringFree( configure );
XmStringFree( execute );
XmStringFree( post );
XmStringFree( help );

// Inform the MenuBar which one of its CascadeButtons contains the "Help" menu
Widget help_menu;
if ( help_menu = XtNameToWidget( menu_bar, "button_4" ) )
  XtVaSetValues( menu_bar, XmNmenuHelpWidget, help_menu, NULL );

// first menu is the "File" menu
XmString connect = XmStringCreateSimple( "Connect..." );
XmString disconnect = XmStringCreateSimple( "Disconnect" );
XmString new_model = XmStringCreateSimple( "New Model..." );
XmString open_model = XmStringCreateSimple( "Open Model..." );
XmString close_model = XmStringCreateSimple( "Close" );
XmString save_as_model = XmStringCreateSimple( "Save As..." );
XmString quit = XmStringCreateSimple( "Exit" );
XmVaCreateSimplePulldownMenu( menu_bar, "file_menu", 0, menu_file_cb,
  XmVaPUSHBUTTON, connect, NULL, NULL, NULL,
  XmVaPUSHBUTTON, disconnect, NULL, NULL, NULL,
  XmVaSEPARATOR,
  XmVaPUSHBUTTON, new_model, 'N', NULL, NULL,
  XmVaPUSHBUTTON, open_model, 'O', NULL, NULL,
  XmVaPUSHBUTTON, close_model, 'C', NULL, NULL,
  XmVaPUSHBUTTON, save_as_model, 'S', NULL, NULL,
  XmVaSEPARATOR,
  XmVaPUSHBUTTON, quit, NULL, NULL, NULL,
  NULL );
XmStringFree( connect );
XmStringFree( disconnect );
XmStringFree( new_model );
XmStringFree( open_model );
XmStringFree( close_model );
XmStringFree( save_as_model );
XmStringFree( quit );

// second menu is the "Configure" menu
XmString create_scenario = XmStringCreateSimple( "Create Scenario..." );
XmString clear_scenario = XmStringCreateSimple( "Clear Scenario" );
XmVaCreateSimplePulldownMenu( menu_bar, "configure_menu", 1,
  menu_configure_cb,
  XmVaPUSHBUTTON, create_scenario, 'C', NULL, NULL,
  XmVaPUSHBUTTON, clear_scenario, 'S', NULL, NULL,
  NULL );
XmStringFree( create_scenario );
XmStringFree( clear_scenario );

// third menu is the "Execute" menu
XmString time_slice = XmStringCreateSimple( "Set Time Slice..." );
XmString time_ratio = XmStringCreateSimple( "Set Time Ratio..." );
XmString run = XmStringCreateSimple( "Run" );
XmString run_until = XmStringCreateSimple( "Run Until..." );
XmVaCreateSimplePulldownMenu( menu_bar, "execute_menu", 2, menu_execute_cb,
  XmVaPUSHBUTTON, time_slice, 'S', NULL, NULL,
  XmVaPUSHBUTTON, time_ratio, 'T', NULL, NULL,
  XmVaSEPARATOR,
  XmVaPUSHBUTTON, run, 'R', NULL, NULL,
  XmVaPUSHBUTTON, run_until, 'U', NULL, NULL,
  NULL );
```

```
XmStringFree( time_slice );
XmStringFree( time_ratio );
XmStringFree( run );
XmStringFree( run_until );

// fourth menu is the "Post Process" menu
XmString view = XmStringCreateSimple( "View Map..." );
XmString reports = XmStringCreateSimple( "Generate Report" );
XmVaCreateSimplePulldownMenu( menu_bar, "post_menu", 3, menu_post_cb,
  XmVaPUSHBUTTON, view, 'V', NULL, NULL,
  XmVaPUSHBUTTON, reports, 'R', NULL, NULL,
  NULL );
XmStringFree( view );
XmStringFree( reports );

// fifth menu is the "Help" menu
XmString about = XmStringCreateSimple( "About SimBench..." );
XmVaCreateSimplePulldownMenu( menu_bar, "help_menu", 4, menu_help_cb,
  XmVaPUSHBUTTON, about, 'A', NULL, NULL,
  NULL );
XmStringFree( about );

XtManageChild( menu_bar );

Widget pane = XtVaCreateWidget( "pane", xmPanedWindowWidgetClass,
  main_window,
  XmNsashWidth, 1,
  XmNsashHeight, 1,
  NULL );

Arg args[9];
XtSetArg( args[0], XmNrows, 20);
XtSetArg( args[1], XmNcolumns, 80);
XtSetArg( args[2], XmNeditable, False);
XtSetArg( args[3], XmNeditMode, XmMULTI_LINE_EDIT);
XtSetArg( args[4], XmNwordWrap, True);
XtSetArg( args[5], XmNscrollHorizontal, False);
XtSetArg( args[6], XmNblinkRate, 0);
XtSetArg( args[7], XmNautoShowCursorPosition, True);
XtSetArg( args[8], XmNcursorPositionVisible, False);
log = XmCreateScrolledText( pane, "log", args, 9 );
XtManageChild( log );

Widget message_frame = XtVaCreateManagedWidget( "message_frame",
  xmFrameWidgetClass, pane,
  XmNshadowType, XmSHADOW_IN,
  NULL );
message = XtVaCreateManagedWidget( "message", xmLabelWidgetClass,
  message_frame,
  XmNalignment, XmALIGNMENT_BEGINNING,
  NULL );
// make sure that the message area can be always be seen
XtWidgetGeometry size;
size.request_mode = CWHeight;
XtQueryGeometry( message_frame, NULL, &size );
XtVaSetValues( message_frame,
  XmNpaneMaximum, size.height,
  XmNpaneMinimum, size.height,
  NULL );
XtManageChild( pane );

XtVaSetValues( main_window,
  XmNmenuBar, menu_bar,
  XmNworkWindow, pane,
```

```
    NULL );
}
////////////////////////////////////////////////////////////////////////
void update_map_dialog( )
{
  int map_width;
  int map_height;
  double elapsed_time;

  if ( map ) {
    // destroy the old map
    XtDestroyWidget( map );
    map = NULL;
  }

  // start the timer to time the map update
  stopwatch map_timer;
  map_timer.start( );

  if ( !sim_get_hexboard_size( map_width, map_height ) ) {
    // failed to determine the size of the hexboard
    generic_ask_user( XmDIALOG_ERROR,
      "Unable to determine the size of the hexboard",
      "OK", NULL, YES );
  }
  else {
    map = XtVaCreateWidget( "hexboard", hexWidgetClass, map_control,
      XmNborderWidth, 0,
      XtNhexRadius, HEX_RADIUS,
      XtNnumberHexX, map_width,
      XtNnumberHexY, map_height,
      NULL );
    sim_add_players_to_hexboard( );
    XtManageChild( map );
  }
  update_model_msg( TRUE );  // update the message area

  // stop the timer
  elapsed_time = map_timer.stop( );
  log_print( "Creating the map took %7.3f seconds\n", elapsed_time );
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////
////  User Information Routines
////
////////////////////////////////////////////////////////////////////////
void log_print( char *fmt, ... )
{
  char buffer[100];
  static XmTextPosition wpr_position;  // maintain text position
  va_list args;

  va_start( args, fmt );
  (void) vsprintf( buffer, fmt, args );
  va_end( args );

  XmTextInsert( log, wpr_position, buffer );
  wpr_position += strlen( buffer );
  XtVaSetValues( log, XmNcursorPosition, wpr_position, NULL );
  XmTextShowPosition( log, wpr_position );
}
////////////////////////////////////////////////////////////////////////
```

```
void msg_print( char *fmt, ... )
{
  char buffer[100];
  va_list args;

  va_start( args, fmt );
  (void) vsprintf( buffer, fmt, args );
  va_end( args );

  XmString string = XmStringCreateSimple( buffer );
  XtVaSetValues( message,
    XmNlabelString, string,
    NULL );
  XmStringFree( string );
}
////////////////////////////////////////////////////////////////////////
void update_model_msg( int model_exists )
{
  if ( model_exists )
    msg_print( "Current model: %s [simulation time %d]",
      sim_get_current_model_name( ), sim_get_current_sim_time( ) );
  else
    msg_print( "Current model: none" );
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////
////  Xt Callback Support Routines
////
////////////////////////////////////////////////////////////////////////
char *get_about_message( )
{
#define SYSTIME_MSG   "System time used:"
#define USRTIME_MSG   "User time used:"
#define MINPFAULT_MSG "Page faults (no physical I/O):"
#define MAJPFAULT_MSG "Page faults (physical I/O):"
#define SWAP_MSG      "Number of program swaps:"
#define MAXSET_MSG    "Maximum resident set size:"
#define SEC_MSG       "seconds"
#define USEC_MSG      "microseconds"
#define BYTE_MSG      "bytes"
  static char buffer[500];
  struct rusage rusage;

  // lookup the system resource usage
  if ( getrusage( RUSAGE_SELF, &rusage ) ) {
    fprintf( stderr, "ERROR[get_about_message] getrusage call failure\n" );
    exit( 1 );
  }
  // update the message
  sprintf( buffer, "%s\n%s\n\n%s %d %s %d %s\n%s %d %s %d %s\n"
    "%s %d\n%s %d\n%s %d\n%s %d %s",
    TITLE_MSG,
    VERSION_MSG,
    USRTIME_MSG,
    rusage.ru_utime.tv_sec, SEC_MSG,
    rusage.ru_utime.tv_usec, USEC_MSG,
    SYSTIME_MSG,
    rusage.ru_stime.tv_sec, SEC_MSG,
    rusage.ru_stime.tv_usec, USEC_MSG,
    MINPFAULT_MSG, rusage.ru_minflt,
    MAJPFAULT_MSG, rusage.ru_majflt,
    SWAP_MSG, rusage.ru_nswap,
```

```
    MAXSET_MSG,
    rusage.ru_maxrss * getpagesize( ), BYTE_MSG );
  return buffer;
}
////////////////////////////////////////////////////////////////////////
Widget get_top_shell( Widget w )
{
  while ( w && !XtIsWMShell( w ) )
    w = XtParent( w );
  return w;
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////
////  Xt Callback Routines
////
////////////////////////////////////////////////////////////////////////
void done_running_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  int remove_work_proc = (int)closure;

  if ( remove_work_proc ) {
    XtWorkProcId work_id;
    XtVaGetValues( w, XmNuserData, &work_id, NULL );
    XtRemoveWorkProc( work_id );
  }
  update_model_msg( TRUE );  // update the message area

  // take down the running dialog
  XtRemoveAllCallbacks( w, XmNcancelCallback );
  XtPopdown( XtParent( w ) );
  XSync( XtDisplay( w ), 0 );
  XmUpdateDisplay( get_top_shell( w ) );
}
////////////////////////////////////////////////////////////////////////
void menu_configure_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  char buffer[100];
  int item_no = (int)closure;
  static Widget create_scenario_dialog = NULL;
  double elapsed_time;

  switch ( item_no ) {
  case 0: // user selected "Create Scenario..."
    if ( !create_scenario_dialog ) {
      create_scenario_dialog = XtVaCreatePopupShell( "create_scenario",
        xmDialogShellWidgetClass, get_top_shell( w ),
        XtNtitle, "Create Scenario",
        NULL );
      Widget pane = XtVaCreateWidget( "pane", xmPanedWindowWidgetClass,
        create_scenario_dialog,
        XmNsashWidth, 1,
        XmNsashHeight, 1,
        NULL );

      // set up the control area of the dialog
      Widget control = XtVaCreateWidget( "control", xmFormWidgetClass, pane,
        NULL );
      XmString prompt_text = XmStringCreateSimple(
        "Enter your desired scenario information:" );
      Widget prompt = XtVaCreateManagedWidget( "prompt", xmLabelWidgetClass,
        control,
        XmNlabelString, prompt_text,
```

```
    XmNtopAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_FORM,
    XmNrightAttachment, XmATTACH_FORM,
    XmNalignment, XmALIGNMENT_BEGINNING,
    NULL );
  XmStringFree( prompt_text );
  Widget subform = prompt;
  Widget label, text;
  for ( int i = 0 ; i < XtNumber( scenario_prompts ) ; i++ ) {
    subform = XtVaCreateWidget( "subform", xmFormWidgetClass,
      control,
      XmNtopAttachment, XmATTACH_WIDGET,
      XmNtopWidget, subform,
      XmNleftAttachment, XmATTACH_FORM,
      XmNrightAttachment, XmATTACH_FORM,
      NULL );
    if ( i == 0 ) XtVaSetValues( subform, XmNtopOffset, 10, NULL );
    label = XtVaCreateManagedWidget( scenario_prompts[i],
      xmLabelWidgetClass, subform,
      XmNtopAttachment, XmATTACH_FORM,
      XmNbottomAttachment, XmATTACH_FORM,
      XmNleftAttachment, XmATTACH_FORM,
      XmNalignment, XmALIGNMENT_BEGINNING,
      NULL );
    sprintf( buffer, "text_%d", i );
    text = XtVaCreateManagedWidget( buffer,
      xmTextWidgetClass, subform,
      XmNmaxLength, 10,
      XmNtopAttachment, XmATTACH_FORM,
      XmNbottomAttachment, XmATTACH_FORM,
      XmNrightAttachment, XmATTACH_FORM,
      XmNleftAttachment, XmATTACH_WIDGET,
      XmNleftWidget, label,
      XmNleftOffset, 20,
      NULL );
    XtManageChild( subform );
  }
  XtManageChild( control );
  // set up the action area of the dialog
  Widget action = XtVaCreateWidget( "action", xmFormWidgetClass, pane,
    XmNfractionBase, 7,
    NULL );
  Widget ok = XtVaCreateManagedWidget( "OK",
    xmPushButtonWidgetClass, action,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_POSITION,
    XmNleftPosition, 1,
    XmNrightAttachment, XmATTACH_POSITION,
    XmNrightPosition, 2,
    XmNshowAsDefault, True,
    XmNdefaultButtonShadowThickness, 1,
    NULL );
  XtAddCallback( ok, XmNactivateCallback,
    try_create_scenario_cb, "create_scenario" );
  Widget cancel = XtVaCreateManagedWidget( "Cancel",
    xmPushButtonWidgetClass, action,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_POSITION,
    XmNleftPosition, 3,
    XmNrightAttachment, XmATTACH_POSITION,
    XmNrightPosition, 4,
    XmNdefaultButtonShadowThickness, 1,
```

```
        NULL );
      XtAddCallback( cancel, XmNactivateCallback,
        take_down_named_cb, "create_scenario" );
      Widget help = XtVaCreateManagedWidget( "Help",
        xmPushButtonWidgetClass, action,
        XmNtopAttachment, XmATTACH_FORM,
        XmNbottomAttachment, XmATTACH_FORM,
        XmNleftAttachment, XmATTACH_POSITION,
        XmNleftPosition, 5,
        XmNrightAttachment, XmATTACH_POSITION,
        XmNrightPosition, 6,
        XmNdefaultButtonShadowThickness, 1,
        NULL );
      XtSetSensitive( help, False );
      XtManageChild( action );
      XtManageChild( pane );
    }
    XtManageChild( create_scenario_dialog );
    XtPopup( create_scenario_dialog, XtGrabNone );
    break;

  case 1: // user selected "Clear Scenario..."
    if ( !sim_configure_clear( elapsed_time ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Can't clear the model's scenario", "OK", NULL, YES );
    }
    else {
      // succeeded
      update_model_msg( TRUE );
      log_print( "Scenario cleared in %7.3f seconds\n", elapsed_time );
      sprintf( buffer, "Scenario cleared in %7.3f seconds", elapsed_time );
      generic_ask_user( XmDIALOG_INFORMATION, buffer, "OK", NULL, YES );
    }
    break;

  default:
    fprintf( stderr, "ERROR[menu_configure_cb] impossible menu choice\n" );
    exit( 1 );
    break;
  }
}
////////////////////////////////////////////////////////////////////////
void menu_execute_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  int item_no = (int)closure;
  static Widget slice_dialog = NULL;
  static Widget ratio_dialog = NULL;
  static Widget run_until_dialog = NULL;

  switch ( item_no ) {
  case 0: // user selected "Set Time Slice..."
    if ( !slice_dialog ) {
      Arg args[3];
      XmString text = XmStringCreateSimple( SLICE_DIALOG_MSG );
      XtSetArg( args[0], XmNselectionLabelString, text );
      XtSetArg( args[1], XmNautoUnmanage, False );
      XtSetArg( args[2], XmNnoResize, True );
      slice_dialog = XmCreatePromptDialog( get_top_shell( w ), "slice",
        args, 3 );
      XmStringFree( text );
      XtVaSetValues( XtParent( slice_dialog ),
        XtNtitle, "Set Time Slice",
        NULL );
```

```
    XtSetSensitive(
      XmSelectionBoxGetChild( slice_dialog, XmDIALOG_HELP_BUTTON ), False );
    XtAddCallback( slice_dialog, XmNokCallback, try_set_time_slice_cb, NULL );
    XtAddCallback( slice_dialog, XmNcancelCallback, take_down_cb, NULL );
  }
  XtManageChild( slice_dialog );
  XtPopup( XtParent( slice_dialog ), XtGrabNone );
  break;

case 1: // user selected "Set Time Ratio..."
  if ( !ratio_dialog ) {
    Arg args[3];
    XmString text = XmStringCreateSimple( RATIO_DIALOG_MSG );
    XtSetArg( args[0], XmNselectionLabelString, text );
    XtSetArg( args[1], XmNautoUnmanage, False );
    XtSetArg( args[2], XmNnoResize, True );
    ratio_dialog = XmCreatePromptDialog( get_top_shell( w ), "ratio",
      args, 3 );
    XmStringFree( text );
    XtVaSetValues( XtParent( ratio_dialog ),
      XtNtitle, "Set Time Ratio",
      NULL );
    XtSetSensitive(
      XmSelectionBoxGetChild( ratio_dialog, XmDIALOG_HELP_BUTTON ), False );
    XtAddCallback( ratio_dialog, XmNokCallback, try_set_time_ratio_cb, NULL );
    XtAddCallback( ratio_dialog, XmNcancelCallback, take_down_cb, NULL );
  }
  XtManageChild( ratio_dialog );
  XtPopup( XtParent( ratio_dialog ), XtGrabNone );
  break;

case 2: // user selected "Run"
  // put up the running dialog
  // set the cancel button to "Stop"
  XmString stop = XmStringCreateSimple( "Stop" );
  XtVaSetValues( running_dialog , XmNcancelLabelString, stop, NULL );
  XmStringFree( stop );
  // set up the work procedure
  XtWorkProcId work_id = XtAppAddWorkProc( app, run_work_proc,
    (XtPointer)running_dialog );
  XtVaSetValues( running_dialog, XmNuserData, work_id, NULL );
  // use cancel button to stop run ("True" means to remove the work proc)
  XtAddCallback( running_dialog, XmNcancelCallback, done_running_cb,
    (XtPointer)True );

  XtManageChild( running_dialog );
  XtPopup( XtParent( running_dialog ), XtGrabNone );

  // set the last run time to the current time
  last_run_clock_time = get_clock_time( );
  break;

case 3: // user selected "Run Until..."
  if ( !run_until_dialog ) {
    Arg args[3];
    XmString text = XmStringCreateSimple( RUN_UNTIL_DIALOG_MSG );
    XtSetArg( args[0], XmNselectionLabelString, text );
    XtSetArg( args[1], XmNautoUnmanage, False );
    XtSetArg( args[2], XmNnoResize, True );
    run_until_dialog = XmCreatePromptDialog( get_top_shell( w ), "run_until",
      args, 3 );
    XmStringFree( text );
    XtVaSetValues( XtParent( run_until_dialog ),
      XtNtitle, "Run Until",
```

```
        NULL );
      XtSetSensitive( XmSelectionBoxGetChild( run_until_dialog,
        XmDIALOG_HELP_BUTTON ), False );
      XtAddCallback( run_until_dialog, XmNokCallback, try_run_until_cb, NULL );
      XtAddCallback( run_until_dialog, XmNcancelCallback, take_down_cb, NULL );
    }
    XtManageChild( run_until_dialog );
    XtPopup( XtParent( run_until_dialog ), XtGrabNone );
    break;

  default:
    fprintf( stderr, "ERROR[menu_execute_cb] impossible menu choice\n" );
    exit( 1 );
    break;
  }
}
////////////////////////////////////////////////////////////////////////
void menu_file_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  int item_no = (int)closure;
  static Widget connect_dialog = NULL;
  static Widget new_model_dialog = NULL;
  static Widget open_model_dialog = NULL;
  static Widget save_as_dialog = NULL;

  switch ( item_no ) {
  case 0: // user selected "Connect..."
    if ( !connect_dialog ) {
      Arg args[3];
      XmString text = XmStringCreateSimple( CONNECT_DIALOG_MSG );
      XtSetArg( args[0], XmNselectionLabelString, text );
      XtSetArg( args[1], XmNautoUnmanage, False );
      XtSetArg( args[2], XmNnoResize, True );
      connect_dialog = XmCreatePromptDialog( get_top_shell( w ), "connect",
        args, 3 );
      XmStringFree( text );
      XtVaSetValues( XtParent( connect_dialog ),
        XtNtitle, "Connect",
        NULL );
      XtSetSensitive(
        XmSelectionBoxGetChild( connect_dialog, XmDIALOG_HELP_BUTTON ), False );
      XtAddCallback( connect_dialog, XmNokCallback, try_connect_cb, NULL );
      XtAddCallback( connect_dialog, XmNcancelCallback, take_down_cb, NULL );
    }
    XtManageChild( connect_dialog );
    XtPopup( XtParent( connect_dialog ), XtGrabNone );
    break;

  case 1: // user selected "Disconnect"
    // attempt to disconnect from the database
    if ( !sim_db_disconnect( ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Can't disconnect from the database", "OK", NULL, YES );
    }
    else {
      // succeeded
      update_model_mag( FALSE );
      log_print( "Successful disconnect from database\n" );

      // Set parts of the the "File" menu sensitive and parts insensitive
      Widget a_menu;
      if ( a_menu = XtNameToWidget( toplevel,  // Connect...
        "main_window.menu_bar.popup_file_menu.file_menu.button_0" ) )
```

```
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Disconnect
      "main_window.menu_bar.popup_file_menu.file_menu.button_1" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Close
      "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Save As...
      "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
      XtSetSensitive( a_menu, False );

    // Set the entire "Configure" menu insensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
      XtSetSensitive( a_menu, False );

    // Set the entire "Execute" menu insensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
      XtSetSensitive( a_menu, False );

    // Set the entire "Post Process" menu insensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
      XtSetSensitive( a_menu, False );
  }
  break;

case 2: // user selected "New Model..."
  if ( !new_model_dialog ) {
    Arg args[3];
    XmString text = XmStringCreateSimple( "Enter new model name:" );
    XtSetArg( args[0], XmNselectionLabelString, text );
    XtSetArg( args[1], XmNautoUnmanage, False );
    XtSetArg( args[2], XmNnoResize, True );
    new_model_dialog = XmCreatePromptDialog( get_top_shell( w ),
      "new_model", args, 3 );
    XmStringFree( text );
    XtVaSetValues( XtParent( new_model_dialog ),
      XtNtitle, "New Model",
      NULL );
    XtSetSensitive( XmSelectionBoxGetChild( new_model_dialog,
      XmDIALOG_HELP_BUTTON ), False );
    XtAddCallback( new_model_dialog, XmNokCallback, try_new_model_cb, NULL );
    XtAddCallback( new_model_dialog, XmNcancelCallback, take_down_cb, NULL );
  }
  XtManageChild( new_model_dialog );
  XtPopup( XtParent( new_model_dialog ), XtGrabNone );
  break;

case 3: // user selected "Open Model..."
  if ( !open_model_dialog ) {
    Arg args[4];
    XmString text = XmStringCreateSimple( "Existing Models" );
    XtSetArg( args[0], XmNlistLabelString, text );
    XtSetArg( args[1], XmNautoUnmanage, False );
    XtSetArg( args[2], XmNmustMatch, True );
    XtSetArg( args[3], XmNnoResize, True );
    open_model_dialog = XmCreateSelectionDialog( get_top_shell( w ),
      "open_model", args, 4 );
    XmStringFree( text );
```

```
    XtVaSetValues( XtParent( open_model_dialog ),
      XtNtitle, "Open Model",
      NULL );
    XtSetSensitive( XmSelectionBoxGetChild( open_model_dialog,
      XmDIALOG_HELP_BUTTON ), False );
    XtUnmanageChild( XmSelectionBoxGetChild( open_model_dialog,
      XmDIALOG_APPLY_BUTTON ) );
    XtAddCallback( open_model_dialog, XmNokCallback, try_open_model_cb, NULL );
    XtAddCallback( open_model_dialog, XmNcancelCallback, take_down_cb, NULL );
  }
  char **models;
  int size;
  if ( !sim_model_list_of_existing( &models, size ) ) {
    generic_ask_user( XmDIALOG_ERROR,
      "Can't determine what models exist",
      "OK", NULL, YES );
  }
  else {
    if ( size == 0 ) {
      generic_ask_user( XmDIALOG_INFORMATION,
        "No models currently exist",
        "OK", NULL, YES );
    }
    else {
      XmString *items = (XmString*)XtMalloc( size * sizeof( XmString ) );
      for ( int i = 0; i < size ; i++ )
        items[i] = XmStringCreateSimple( models[i] );
      XtVaSetValues( open_model_dialog,
        XmNlistItems, items,
        XmNlistItemCount, size,
        NULL );
      while ( --i >= 0 )
        XmStringFree( items[i] );  // free elements of the array
      XtFree( (char *)items );  // now free the array pointer
      XtManageChild( open_model_dialog );
      XtPopup( XtParent( open_model_dialog ), XtGrabNone );
    }
  }
  break;

case 4: // user selected "Close"
  // attempt to close the current model
  if ( !sim_model_close( ) ) {
    // failed
    generic_ask_user( XmDIALOG_ERROR,
      "Can't close the current model", "OK", NULL, YES );
  }
  else {
    // succeeded
    update_model_msg( FALSE );
    log_print( "Current model closed\n" );

    // Set parts of the the "File" menu sensitive and parts insensitive
    Widget a_menu;
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Close
      "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Save As...
```

```
        "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
        XtSetSensitive( a_menu, False );

      // Set the entire "Configure" menu insensitive
      if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
        XtSetSensitive( a_menu, False );

      // Set the entire "Execute" menu insensitive
      if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
        XtSetSensitive( a_menu, False );

      // Set the entire "Post Process" menu insensitive
      if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
        XtSetSensitive( a_menu, False );
    }
    break;

  case 5: // user selected "Save As..."
    if ( !save_as_dialog ) {
      Arg args[3];
      XmString text = XmStringCreateSimple( "Enter model name to save as:" );
      XtSetArg( args[0], XmNselectionLabelString, text );
      XtSetArg( args[1], XmNautoUnmanage, False );
      XtSetArg( args[2], XmNnoResize, True );
      save_as_dialog = XmCreatePromptDialog( get_top_shell( w ),
        "save_as", args, 3 );
      XmStringFree( text );
      XtVaSetValues( XtParent( save_as_dialog ),
        XtNtitle, "Save As",
        NULL );
      XtSetSensitive( XmSelectionBoxGetChild( save_as_dialog,
        XmDIALOG_HELP_BUTTON ), False );
      XtAddCallback( save_as_dialog, XmNokCallback, try_save_as_cb, NULL );
      XtAddCallback( save_as_dialog, XmNcancelCallback, take_down_cb, NULL );
    }
    XtManageChild( save_as_dialog );
    XtPopup( XtParent( save_as_dialog ), XtGrabNone );
    break;

  case 6: // user selected "Exit"
    // perform any required database cleanup then exit
    if ( !sim_db_disconnect( ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Can't exit program: the disconnect from the database failed",
        "OK", NULL, YES );
    }
    else
      exit( 1 );
    break;

  default:
    fprintf( stderr, "ERROR[menu_file_cb] impossible menu choice\n" );
    exit( 1 );
    break;
  }
}
//////////////////////////////////////////////////////////////////////
void menu_help_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  static Widget about_dialog = NULL;
  int item_no = (int)closure;
  XmString text;
```

```
  switch ( item_no ) {
  case 0: // user selected "About SimBench..."
    if ( !about_dialog ) {
      Arg args[1];
      XtSetArg( args[0], XmNnoResize, True );
      about_dialog = XmCreateInformationDialog( get_top_shell( w ), "about",
        args, 1 );
      XtVaSetValues( XtParent( about_dialog ),
        XtNtitle, "About SimBench",
        NULL );
      XtSetSensitive(
        XmMessageBoxGetChild( about_dialog, XmDIALOG_HELP_BUTTON ), False );
      XtUnmanageChild(
        XmMessageBoxGetChild( about_dialog, XmDIALOG_CANCEL_BUTTON ) );
    }
    text = XmStringCreateLtoR( get_about_message( ), XmSTRING_DEFAULT_CHARSET );
    XtVaSetValues( about_dialog,
      XmNmessageString, text,
      NULL );
    XmStringFree( text );
    XtManageChild( about_dialog );
    XtPopup( XtParent( about_dialog ), XtGrabNone );
    break;

  default:
    fprintf( stderr, "ERROR[menu_help_cb] impossible menu choice\n" );
    exit( 1 );
    break;
  }
}
////////////////////////////////////////////////////////////////////////
void menu_post_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  char buffer[100];
  int item_no = (int)closure;
  double elapsed_time;

  switch ( item_no ) {
  case 0: // user selected "View Map..."
    if ( !map_dialog ) {
      // create the map dialog the first time this menu choice is selected
      create_map_dialog( );
    }
    update_map_dialog( );
    XtManageChild( map_dialog );
    XtPopup( map_dialog, XtGrabNone );
    break;

  case 1: // user selected "Generate Report"
    if ( !sim_report( elapsed_time ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Generation of report failed", "OK", NULL, YES );
    }
    else {
      // succeeded
      log_print( "Report generated in %7.3f seconds\n", elapsed_time );
      sprintf( buffer,
        "Report generated in %7.3f seconds (to file \"SIM_REPORT\")",
        elapsed_time );
      generic_ask_user( XmDIALOG_INFORMATION, buffer, "OK", NULL, YES );
    }
    break;
```

```
  default:
    fprintf( stderr, "ERROR[menu_help_cb] impossible menu choice\n" );
    exit( 1 );
    break;
  }
}
////////////////////////////////////////////////////////////////////////
void take_down_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XtPopdown( XtParent( w ) );
  XSync( XtDisplay( w ), 0 );
  XmUpdateDisplay( get_top_shell( w ) );
}
////////////////////////////////////////////////////////////////////////
void take_down_named_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  Widget dialog;
  char *dialog_name = (char*)closure;

  if ( dialog = XtNameToWidget( toplevel, dialog_name ) ) {
    XtPopdown( dialog );
    XSync( XtDisplay( dialog ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );
  }
}
////////////////////////////////////////////////////////////////////////
void try_connect_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *db_name;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET, &db_name ) ) {
    fprintf( stderr, "ERROR[try_connect_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }

  // attempt to connect to the database
  if ( !sim_db_connect( db_name ) ) {
    // failed
    generic_ask_user( XmDIALOG_ERROR,
      "Can't connect to the database you have specified",
      "OK", NULL, YES );
  }
  else {
    // succeeded
    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );

    log_print( "Successful connect to database \"%s\"\n", db_name );

    // Set parts of the the "File" menu sensitive and parts insensitive
    Widget a_menu;
    if ( a_menu = XtNameToWidget( toplevel,  // Connect...
      "main_window.menu_bar.popup_file_menu.file_menu.button_0" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Disconnect
      "main_window.menu_bar.popup_file_menu.file_menu.button_1" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
```

```
    XtSetSensitive( a_menu, True );
  }
  XtFree( db_name );
}
////////////////////////////////////////////////////////////////////////
void try_create_scenario_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  char buffer[100];
  Widget dialog, text_0, text_1, text_2;

  // get the values of the three text widgets
  if ( !( text_0 = XtNameToWidget( toplevel,
    "create_scenario.pane.control.subform.text_0" ) ) ) {
    fprintf( stderr, "ERROR[try_create_scenario_cb] unable to find text_0\n" );
    exit( 1 );
  }
  if ( !( text_1 = XtNameToWidget( toplevel,
    "create_scenario.pane.control.subform.text_1" ) ) ) {
    fprintf( stderr, "ERROR[try_create_scenario_cb] unable to find text_1\n" );
    exit( 1 );
  }
  if ( !( text_2 = XtNameToWidget( toplevel,
    "create_scenario.pane.control.subform.text_2" ) ) ) {
    fprintf( stderr, "ERROR[try_create_scenario_cb] unable to find text_2\n" );
    exit( 1 );
  }

  // try to get the numbers input by the user
  char *num_aircraft_char = XmTextGetString( text_0 );
  int num_aircraft = atoi( num_aircraft_char );
  XtFree( num_aircraft_char );
  char *num_trucks_char = XmTextGetString( text_1 );
  int num_trucks = atoi( num_trucks_char );
  XtFree( num_trucks_char );
  char *hexboard_size_char = XmTextGetString( text_2 );
  int hexboard_size = atoi( hexboard_size_char );
  XtFree( hexboard_size_char );

  // check that the values are valid
  // (if any of the values are zero the input is invalid)
  if ( ( num_aircraft < 1 )||( num_aircraft > 100000 )||
    ( num_trucks < 1 )||( num_trucks > 100000 )||
    ( hexboard_size < 1 )||( hexboard_size > 10000 ) ) {
    // failed - invalid input
    generic_ask_user( XmDIALOG_ERROR,
      "The values you have specified are invalid",
      "OK", NULL, YES );
  }
  else {
    double elapsed_time;
    if ( !sim_configure_create( num_aircraft, num_trucks, hexboard_size,
      elapsed_time ) ) {
    // failed - database problem
    generic_ask_user( XmDIALOG_ERROR,
      "Unable to create the scenario in the database",
      "OK", NULL, YES );
    }
    else {
      // succeeded
      update_model_msg( TRUE );
      log_print( "Scenario created in %7.3f seconds\n", elapsed_time );
      sprintf( buffer, "Scenario created in %7.3f seconds", elapsed_time );
      generic_ask_user( XmDIALOG_INFORMATION, buffer,
        "OK", NULL, YES );
```

```
      // popdown the create scenario dialog
      if ( dialog = XtNameToWidget( toplevel, "create_scenario" ) ) {
        XtPopdown( dialog );
        XSync( XtDisplay( dialog ), 0 );
        XmUpdateDisplay( get_top_shell( w ) );
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////
void try_new_model_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *model_name;
  double elapsed_time;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET, &model_name ) ) {
    fprintf( stderr, "ERROR[try_new_model_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }

  // attempt to create a new model
  if ( !sim_model_new( model_name, elapsed_time ) ) {
    // failed
    generic_ask_user( XmDIALOG_ERROR,
      "Can't create the model you have specified",
      "OK", NULL, YES );
  }
  else {
    // succeeded
    update_model_msg( TRUE );
    log_print( "Model \"%s\" created in %7.3f seconds\n",
      model_name, elapsed_time );

    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );

    // Set parts of the the "File" menu sensitive and parts insensitive
    Widget a_menu;
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Close
      "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Save As...
      "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Configure" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Execute" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Post Process" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
```

```
    XtSetSensitive( a_menu, True );
  }
  XtFree( model_name );
}
////////////////////////////////////////////////////////////////////////
void try_open_model_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *model_name;
  double elapsed_time;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET, &model_name ) ) {
    fprintf( stderr, "ERROR[try_open_model_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }

  // attempt to open the specified model
  if ( !sim_model_open( model_name, elapsed_time ) ) {
    // failed
    generic_ask_user( XmDIALOG_ERROR,
      "Can't open the model you have specified",
      "OK", NULL, YES );
  }
  else {
    // succeeded
    update_model_msg( TRUE );
    log_print( "Model \"%s\" opened in %7.3f seconds\n",
      model_name, elapsed_time );

    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );

    // Set parts of the the "File" menu sensitive and parts insensitive
    Widget a_menu;
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Close
      "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Save As...
      "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Configure" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Execute" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Post Process" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
      XtSetSensitive( a_menu, True );
  }
  XtFree( model_name );
}
////////////////////////////////////////////////////////////////////////
void try_run_until_cb( Widget w, XtPointer closure, XtPointer call_data )
```

```
{
  char buffer[100];
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  int current_sim_time;
  char *goal_time_char;
  int goal_time;

  // read the desired goal time
  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET,
    &goal_time_char ) ) {
    fprintf( stderr, "ERROR[try_run_until_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }
  goal_time = atoi( goal_time_char );
  XtFree( goal_time_char );

  // check the current simulation time compaired to the goal time
  current_sim_time = sim_get_current_sim_time( );
  if ( goal_time <= current_sim_time ) {
    // can't run the simulation backwards
    sprintf( buffer, "Can't simulate in reverse (at simulation tim  %d)",
      current_sim_time );
    generic_ask_user( XmDIALOG_ERROR, buffer , "OK", NULL, YES );
  }
  else {
    // set up to perform the run until
    run_until_goal_time = goal_time;

    // take down the run until dialog
    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );

    // put up the running dialog
    // set the cancel button to "Stop"
    XmString stop = XmStringCreateSimple( "Stop" );
    XtVaSetValues( running_dialog , XmNcancelLabelString, stop, NULL );
    XmStringFree( stop );
    // set up the work procedure
    XtWorkProcId work_id = XtAppAddWorkProc( app, run_until_work_proc,
      (XtPointer)running_dialog );
    XtVaSetValues( running_dialog, XmNuserData, work_id, NULL );
    // use cancel button to stop run ("True" means to remove the work proc)
    XtAddCallback( running_dialog, XmNcancelCallback, done_running_cb,
      (XtPointer)True );

    XtManageChild( running_dialog );
    XtPopup( XtParent( running_dialog ), XtGrabNone );

    // start the timer for the run until
    global_timer = new stopwatch( );
    global_timer->start( );
  }
}
////////////////////////////////////////////////////////////////////////
void try_save_as_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *model_name;
  double elapsed_time;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET, &model_name ) ) {
    fprintf( stderr, "ERROR[try_save_as_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
```

```
  }

  // attempt to save the current model under "model_name"
  if ( !sim_model_save_as( model_name, elapsed_time ) ) {
    // failed
    generic_ask_user( XmDIALOG_ERROR,
      "Can't save the current model as the model you have specified",
      "OK", NULL, YES );
  }
  else {
    // succeeded
    log_print( "Current model copied to \"%s\" in %7.3f seconds\n",
      model_name, elapsed_time );

    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );

    // Set parts of the the "File" menu sensitive and parts insensitive
    Widget a_menu;
    if ( a_menu = XtNameToWidget( toplevel,  // New Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_2" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Open Model...
      "main_window.menu_bar.popup_file_menu.file_menu.button_3" ) )
      XtSetSensitive( a_menu, False );
    if ( a_menu = XtNameToWidget( toplevel,  // Close
      "main_window.menu_bar.popup_file_menu.file_menu.button_4" ) )
      XtSetSensitive( a_menu, True );
    if ( a_menu = XtNameToWidget( toplevel,  // Save As...
      "main_window.menu_bar.popup_file_menu.file_menu.button_5" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Configure" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_1" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Execute" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_2" ) )
      XtSetSensitive( a_menu, True );

    // Set the entire "Post Process" menu sensitive
    if ( a_menu = XtNameToWidget( toplevel, "main_window.menu_bar.button_3" ) )
      XtSetSensitive( a_menu, True );
  }
  XtFree( model_name );
}
////////////////////////////////////////////////////////////////////////
void try_set_time_ratio_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *new_time_ratio_char;
  double new_time_ratio;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET,
    &new_time_ratio_char ) ) {
    fprintf( stderr, "ERROR[try_set_time_ratio_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }
  new_time_ratio = atof( new_time_ratio_char );
  XtFree( new_time_ratio_char );

  if ( new_time_ratio <= 0.0 ) {
    // invalid time ratio
```

```
    generic_ask_user( XmDIALOG_ERROR,
      "The time ratio you have specified is invalid",
      "OK", NULL, YES );
  }
  else {
    // set the time ratio
    sim_set_time_ratio( new_time_ratio );

    log_print( "Time ratio set to %f\n", sim_get_time_ratio( ) );

    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );
  }
}
////////////////////////////////////////////////////////////////////////
void try_set_time_slice_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  XmSelectionBoxCallbackStruct *cbs = (XmSelectionBoxCallbackStruct*)call_data;
  char *new_time_slice_char;
  int new_time_slice;

  if ( !XmStringGetLtoR( cbs->value, XmSTRING_DEFAULT_CHARSET,
    &new_time_slice_char ) ) {
    fprintf( stderr, "ERROR[try_set_time_slice_cb] XmStringGetLtoR failed\n" );
    exit( 1 );
  }
  new_time_slice = atoi( new_time_slice_char );
  XtFree( new_time_slice_char );

  if ( new_time_slice < 1 ) {
    // invalid time slice
    generic_ask_user( XmDIALOG_ERROR,
      "The time slice you have specified is invalid",
      "OK", NULL, YES );
  }
  else {
    // set the time slice
    sim_set_time_slice( new_time_slice );

    log_print( "Time slice set to %d\n", sim_get_time_slice( ) );

    XtPopdown( XtParent( w ) );
    XSync( XtDisplay( w ), 0 );
    XmUpdateDisplay( get_top_shell( w ) );
  }
}
////////////////////////////////////////////////////////////////////////
void update_map_cb( Widget w, XtPointer closure, XtPointer call_data )
{
  // check to ensure that there is a current model
  if ( !sim_get_current_model_name( ) ) {
    // there is no current model
    generic_ask_user( XmDIALOG_ERROR,
      "The current model has been closed, so the map can't be updated",
      "OK", NULL, YES );
  }
  else {
    // call the function to update the map dialog
    update_map_dialog( );
  }
}
////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////
////
////  It Work Procedures
////
////////////////////////////////////////////////////////////////////////
Boolean run_until_work_proc( XtPointer client_data )
{
  Arg args[2];
  int n;
  char buffer[100];
  Widget running_dialog = (Widget)client_data;
  int stop_work_proc = FALSE;

  // get the current simulation time
  int current_sim_time = sim_get_current_sim_time( );

  // set up labels
  sprintf( buffer, "Running to time %d...at time %d", run_until_goal_time,
    current_sim_time );
  XmString message = XmStringCreateSimple( buffer );
  XmString button = XmStringCreateSimple( "Done" );
  n = 0;
  XtSetArg( args[n], XmNmessageString, message ); n++;
  if ( current_sim_time >= run_until_goal_time ) {
    // the simulation run is done

    // stop the run until timer
    double elapsed_time = global_timer->stop( );
    delete global_timer;
    log_print( "Running until simulation time %d took %7.3f seconds\n",
      sim_get_current_sim_time( ), elapsed_time );

    stop_work_proc = TRUE;
    XtSetArg( args[n], XmNcancelLabelString, button ); n++;
    XtRemoveAllCallbacks( running_dialog, XmNcancelCallback );
    XtAddCallback( running_dialog, XmNcancelCallback, done_running_cb,
      (XtPointer)False );
    // "ring" the user
    XBell( XtDisplay( running_dialog ), 50 );
  }
  else {
    // execute a time slice of simulation time
    int step_sim_goal_time = current_sim_time + sim_get_time_slice( );
    if ( step_sim_goal_time > run_until_goal_time )
      step_sim_goal_time = run_until_goal_time;
    if ( !sim_run_until_sim_goal_time( step_sim_goal_time ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Simulation failed execution",
        "OK", NULL, YES );
    }
  }
  XtSetValues( running_dialog, args, n );
  XmStringFree( message );
  XmStringFree( button );

  // return either TRUE (we're done, remove the work proc)
  //  or FALSE (continue working by calling this function)
  return stop_work_proc;
}
////////////////////////////////////////////////////////////////////////
Boolean run_work_proc( XtPointer client_data )
{
  // only do anything if it is time to run a time slice
```

```
  double current_clock_time = get_clock_time( );
  if ( current_clock_time >= next_clock_time_to_run ) {
    Widget running_dialog = (Widget)client_data;

    // read the time slice and time ratio
    int time_slice = sim_get_time_slice( );
    double time_ratio = sim_get_time_ratio( );

    // run the simulation through a time slice
    int current_sim_time = sim_get_current_sim_time( );
    if ( !sim_run_until_sim_goal_time( current_sim_time + time_slice ) ) {
      // failed
      generic_ask_user( XmDIALOG_ERROR,
        "Simulation failed execution",
        "OK", NULL, YES );
    }
    double clock_time_after_run = get_clock_time( );

    // calculate run timing statistics
    double duration_of_run = current_clock_time - last_run_clock_time;
    double duration_of_run_in_sim = clock_time_after_run - current_clock_time;
    double actual_time_ratio = duration_of_run / (double)time_slice;
    last_run_clock_time = current_clock_time;

    // change dialog label
    char buffer[400];
    sprintf( buffer, "Running...at time %d\n"
      "Desired time ratio: %7.3f\n"
      "Actual time ratio: %7.3f\n"
      "Total time duration: %8.4f seconds\n"
      "Total time duration simulating: %8.4f seconds",
      sim_get_current_sim_time( ),
      time_ratio, actual_time_ratio,
      duration_of_run, duration_of_run_in_sim );
    XmString message = XmStringCreateLtoR( buffer, XmSTRING_DEFAULT_CHARSET );
    XtVaSetValues( running_dialog, XmNmessageString, message, NULL );
    XmStringFree( message );

    // calculate next clock time to run
    next_clock_time_to_run = current_clock_time +
      (double)time_slice * time_ratio;
  }

  // always continue work procedure until stopped by user
  return FALSE;
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////
////  Generic Ask User Routines
////
////////////////////////////////////////////////////////////////////////
static void generic_ask_user_response_cb( Widget w, XtPointer closure,
  XtPointer call_data )
{
  XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*)call_data;
  int *answer = (int*)closure;

  switch ( cbs->reason ) {
  case XmCR_OK:
    *answer = YES;
    break;
  case XmCR_CANCEL:
```

```
    *answer = NO;
    break;
  }
}
////////////////////////////////////////////////////////////////////
int generic_ask_user( unsigned char dialog_type, char *question,
  char *answer_yes, char *answer_no, int default_answer )
{
  static Widget dialog = NULL;
  static int answer;

  if ( !dialog ) {
    Arg args[2];
    XtSetArg( args[0], XmNdialogStyle, XmDIALOG_FULL_APPLICATION_MODAL );
    XtSetArg( args[1], XmNnoResize, True );
    dialog = XmCreateMessageDialog( toplevel, "dialog", args, 2 );
    XtSetSensitive( XmMessageBoxGetChild( dialog,
      XmDIALOG_HELP_BUTTON ), False );
    XtAddCallback( dialog, XmNokCallback, generic_ask_user_response_cb,
      (XtPointer)&answer );
    XtAddCallback( dialog, XmNcancelCallback, generic_ask_user_response_cb,
      (XtPointer)&answer );
  }
  answer = 0;
  XmString text = XmStringCreateSimple( question );
  XmString yes = XmStringCreateSimple( answer_yes );
  XmString no = XmStringCreateSimple( answer_no );
  XtVaSetValues( dialog,
    XmNdialogType, dialog_type,
    XmNmessageString, text,
    XmNokLabelString, yes,
    XmNcancelLabelString, no,
    XmNdefaultButtonType, default_answer==YES?
      XmDIALOG_OK_BUTTON : XmDIALOG_CANCEL_BUTTON,
    NULL );
  XmStringFree( text );
  XmStringFree( yes );
  XmStringFree( no );
  // only show the cancel button if "answer_no" is not NULL
  if ( answer_no )
    XtManageChild( XmMessageBoxGetChild( dialog, XmDIALOG_CANCEL_BUTTON ) );
  else
    XtUnmanageChild( XmMessageBoxGetChild( dialog, XmDIALOG_CANCEL_BUTTON ) );
  // set the dialog title based upon the dialog type
  switch ( dialog_type ) {
    case XmDIALOG_ERROR:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Error", NULL );
      break;
    case XmDIALOG_INFORMATION:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Information", NULL );
      break;
    case XmDIALOG_MESSAGE:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Message", NULL );
      break;
    case XmDIALOG_QUESTION:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Question", NULL );
      break;
    case XmDIALOG_WARNING:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Warning", NULL );
      break;
    case XmDIALOG_WORKING:
      XtVaSetValues( XtParent( dialog ), XtNtitle, "Working", NULL );
      break;
    default:
```

```
      fprintf( stderr, "ERROR[generic_ask_user] impossible dialog type\n" );
      exit( 1 );
      break;
  }
  XtManageChild( dialog );
  XtPopup( XtParent( dialog ), XtGrabNone );

  // while the user hasn't provided an answer, simulate the main loop
  // (note that the mode is XmDIALOG_FULL_APPLICATION_MODAL so the
  //  should not notice this "takeover" of the program)
  while ( answer == 0 )
    XtAppProcessEvent( app, XtIMAll );

  XtPopdown( XtParent( dialog ) );
  XSync( XtDisplay( dialog ), 0 );
  XmUpdateDisplay( toplevel );

  return answer;
}
//////////////////////////////////////////////////////////////////////
```

3.1.21 simulate.hh

The source code for the file *simulate.hh* is listed below.

```
#ifndef ___SIMULATE_HH
#define ___SIMULATE_HH
/*

 #####
#     #    #    #    #  #    #  #         ##     #####  ######
#          #    ##  ##  #    #  #        #  #      #    #
 #####     #    # ## #  #    #  #       #    #     #    #####
      #    #    #    #  #    #  #       ######     #    #
#     #    #    #    #  #    #  #       #    #     #    #
 #####     #    #    #   ####   ######  #    #     #    ######

 simulate.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 15 Aug 1993
*/
#define TRUE  1
#define FALSE 0
void sim_add_players_to_hexboard( );
int sim_configure_clear( double& elapsed_time );
int sim_configure_create( int num_aircraft, int num_trucks, int hexboard_size,
  double& elapsed_time );
int sim_db_connect( char* db_name );
int sim_db_disconnect( );
char *sim_get_current_model_name( );
int sim_get_current_sim_time( );
int sim_get_hexboard_size( int& hexboard_width, int& hexboard_height );
double sim_get_time_ratio( );
int sim_get_time_slice( );
int sim_model_close( );
int sim_model_list_of_existing( char ***models, int& size);
int sim_model_new( char* scenario_name, double& elapsed_time );
int sim_model_open( char* scenario_name, double& elapsed_time );
int sim_model_save_as( char* scenario_name, double& elapsed_time );
int sim_report( double& elapsed_time );
int sim_run_until_sim_goal_time( int goal_sim_time );
void sim_set_time_ratio( double new_time_ratio );
```

```
void sim_set_time_slice( int new_time_slice );
#endif ___SIMULATE_HH
```

3.1.22 simulate.cc

The source code for the file *simulate.cc* is listed below.

```
/*

 #####
#     #    #    #    #  #    #  #         ##     #####  ######
#          #    ##  ##  #    #  #        #  #      #    #
 #####     #    # ## #  #    #  #       #    #     #    #####
      #    #    #    #  #    #  #       ######     #    #
#     #    #    #    #  #    #  #       #    #     #    #
 #####     #    #    #   ####   ######  #    #     #    ######

 simulate.cc

 This module controls interactions between the Motif user interface
  (and hence the user) and the object-oriented DBMS.

 Air Force Institute of Technology
 Timothy J. Halloran
 15 Aug 1993
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<Xm/Xm.h>  // only needed for Xt memory management (i.e. XtMalloc)
#include<stdio.h>
#include<stdlib.h>
#include<debug.hh>
#include"hexplay.hh"
#include"location.hh"
#include"stopwtch.hh"
#include"dice.hh"
#include"expon.hh"
#include"simulate.hh"
#include"model.hh"
#include"event.hh"
#include"environ.hh"
#include"player.hh"
#include"hexboard.hh"
#include"aircraft.hh"
#include"truck.hh"

#define TRUE  1
#define FALSE 0

#define STREAM 1
#define FETCH_SIZE_IN_BYTES 8192

static char *aircraft_home_bases[10] = {
  "HOME-BASE0", "HOME-BASE1", "HOME-BASE2", "HOME-BASE3", "HOME-BASE4",
  "HOME-BASE5", "HOME-BASE6", "HOME-BASE7", "HOME-BASE8", "HOME-BASE9"
};

static char *truck_types[10] = {
  "TRUCK-TYPE0", "TRUCK-TYPE1", "TRUCK-TYPE2", "TRUCK-TYPE3", "TRUCK-TYPE4",
  "TRUCK-TYPE5", "TRUCK-TYPE6", "TRUCK-TYPE7", "TRUCK-TYPE8", "TRUCK-TYPE9"
};

static char *truck_payloads[10] = {
```

```
  "CARGO-TYPE0", "CARGO-TYPE1", "CARGO-TYPE2", "CARGO-TYPE3", "CARGO-TYPE4",
  "CARGO-TYPE5", "CARGO-TYPE6", "CARGO-TYPE7", "CARGO-TYPE8", "CARGO-TYPE9"
};

static char *misc_sim_data[10] = {
  "00000-SIM-00000-SIM-00000-SIM-00000-SIM-00000-SIM-",
  "11111-SIM-11111-SIM-11111-SIM-11111-SIM-11111-SIM-",
  "22222-SIM-22222-SIM-22222-SIM-22222-SIM-22222-SIM-",
  "33333-SIM-33333-SIM-33333-SIM-33333-SIM-33333-SIM-",
  "44444-SIM-44444-SIM-44444-SIM-44444-SIM-44444-SIM-",
  "55555-SIM-55555-SIM-55555-SIM-55555-SIM-55555-SIM-",
  "66666-SIM-66666-SIM-66666-SIM-66666-SIM-66666-SIM-",
  "77777-SIM-77777-SIM-77777-SIM-77777-SIM-77777-SIM-",
  "88888-SIM-88888-SIM-88888-SIM-88888-SIM-88888-SIM-",
  "99999-SIM-99999-SIM-99999-SIM-99999-SIM-99999-SIM-"
};

// declare functions used in this module
static model *get_current_model( );

// global and module variables
       os_database *db = NULL;  // required to be here by ObjectStore
       persistent<db> os_workspace *sim_ws = NULL;
       persistent<db> os_Set<model*> *model::extent = 0;
static char *current_model_name = NULL;
static int connected_to_db = FALSE;
static int time_slice = 1;
static double time_ratio = 1.00;
////////////////////////////////////////////////////////////////////////
static model *get_current_model( )
{
  DEBUG_INIT( "SIMULATE" , "get_current_model" );
  DEBUG_OUT( "entering", 1 );

  DEBUG_OUT( sprintf( BUG, "looking up model \"%s\"", current_model_name ), 2 );
  // uses the global variable "current_model_name" to get a pointer to
  //  the current model in the database
  //  (the database must be within a transaction to call this function)
  return ( *model::extent )[%strcmp(name,current_model_name)==0%];
}
////////////////////////////////////////////////////////////////////////
void sim_add_players_to_hexboard( )
{
  DEBUG_INIT( "SIMULATE" , "sim_add_players_to_hexboard" );
  DEBUG_OUT( "entering", 1 );
  int pos_x, pos_y;

  do_transaction () {
    model *current_model = get_current_model( );

    // add all the simulation players to the hexmap
    for ( environment *m = current_model->first_member ; m ; m = m->next ) {
      if ( ( !strcmp( m->query( "CLASS" ), "AIRCRAFT" ) )||
          ( !strcmp( m->query( "CLASS" ), "TRUCK" ) ) ) {
        decode_location( m->query( "LOCATION" ), pos_x, pos_y );
        add_player_to_hexboard( m->query( "CLASS" ), m->query( "NAME" ),
          pos_x, pos_y );
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////
int sim_configure_clear( double& elapsed_time )
{
```

```
  DEBUG_INIT( "SIMULATE" , "sim_configure_clear" );
  DEBUG_OUT( "entering", 1 );

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    model *current_model = get_current_model( );
    // clear out the current model
    current_model->clear( );
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_configure_create( int num_aircraft, int num_trucks, int hexboard_size,
  double& elapsed_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_configure_create" );
  DEBUG_OUT( "entering", 1 );
  int i;
  char new_name[10];
  char new_location[10];

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    model *current_model = get_current_model( );
    // clear out the current model (just in case the user forgot to)
    current_model->clear( );
    // create the hexboard
    current_model->add_environment_member( new(db, current_model->model_config)
      hexboard( "HEXBOARD", hexboard_size, hexboard_size ) );

    // create the trucks
    for ( i = 0 ; i < num_trucks ; i++ ) {
      sprintf( new_name, "GND-%05d", i );
      char *new_type = truck_types[roll( 0, 9, STREAM )];
      char *new_payload = truck_payloads[roll( 0, 9, STREAM )];
      char *new_misc_sim_data = misc_sim_data[roll( 0, 9, STREAM )];
      encode_location( new_location,
        (int)roll( 1, hexboard_size, STREAM ),
        (int)roll( 1, hexboard_size, STREAM ) );
      // create a single truck and add it to the model's environment
      truck *new_truck = new(db, current_model->model_config)
        truck( new_name, new_type, new_payload,
        new_location, new_misc_sim_data );
      current_model->add_environment_member( new_truck );
      // schedule an initial "MOVE" event for the truck
      current_model->schedule( new_truck, new_name,
        current_model->sim_time + (int)expon( 600.0 , STREAM ), "MOVE" );
    }

    // create the aircraft
    for ( i = 0 ; i < num_aircraft ; i++ ) {
      sprintf( new_name, "AIR-%05d", i );
```

```
      char *new_home_base = aircraft_home_bases[roll( 0, 9, STREAM )];
      char *new_misc_sim_data = misc_sim_data[roll( 0, 9, STREAM )];
      encode_location( new_location,
        (int)roll( 1, hexboard_size, STREAM ),
        (int)roll( 1, hexboard_size, STREAM ) );
      // create a single aircraft and add it to the model's environment
      aircraft *new_aircraft = new(db, current_model->model_config)
        aircraft( new_name, new_home_base, new_location, new_misc_sim_data );
      current_model->add_environment_member( new_aircraft );
      // schedule an initial "MOVE" event for the aircraft
      current_model->schedule( new_aircraft, new_name,
        current_model->sim_time + (int)expon( 60.0 , STREAM ), "MOVE" );
    }

  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_db_connect( char* db_name )
{
  DEBUG_INIT( "SIMULATE" , "sim_db_connect" );
  DEBUG_OUT( "entering", 1 );

  objectstore::initialize( );
  os_collection::initialize( );

  TIX_HANDLE(all_exceptions)
    // open the database
    db = os_database::open( db_name );

    // tell ObjectStore not to read the entire segment
    db->set_read_whole_segment( FALSE );

    // set the fetch policy of ObjectStore to fetch a number of bytes at a time
    db->set_fetch_policy( os_fetch_page, FETCH_SIZE_IN_BYTES );

    do_transaction () {
      os_workspace::set_current( sim_ws );
    }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  connected_to_db = TRUE;
  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_db_disconnect( )
{
  DEBUG_INIT( "SIMULATE" , "sim_db_disconnect" );
  DEBUG_OUT( "entering", 1 );
  if ( !connected_to_db ) return TRUE;

  // make sure there is no current model
  if ( current_model_name ) {
    free( current_model_name );
    current_model_name = NULL;
  }
```

```
  TIX_HANDLE(all_exceptions)
    // close the database
    db->close( );
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  connected_to_db = FALSE;
  return TRUE;
}
////////////////////////////////////////////////////////////////////////
char *sim_get_current_model_name( )
{
  return current_model_name;
}
////////////////////////////////////////////////////////////////////////
int sim_get_current_sim_time( )
{
  DEBUG_INIT( "SIMULATE" , "sim_get_current_sim_time" );
  DEBUG_OUT( "entering", 1 );
  int current_sim_time;

  do_transaction () {
    model *current_model = get_current_model( );
    current_sim_time = current_model->sim_time;
  }

  return current_sim_time;
}
////////////////////////////////////////////////////////////////////////
int sim_get_hexboard_size( int& hexboard_width, int& hexboard_height )
{
  DEBUG_INIT( "SIMULATE" , "sim_get_hexboard_size" );
  DEBUG_OUT( "entering", 1 );
  int found_hexboard = FALSE;

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    model *current_model = get_current_model( );

    // determine the size of the hexmap
    for ( environment *m = current_model->first_member ; m ; m = m->next ) {
      if ( !strcmp( m->query( "NAME" ), "HEXBOARD" ) ) {
        hexboard_width = atoi( m->query( "WIDTH" ) );
        hexboard_height = atoi( m->query( "HEIGHT" ) );
        found_hexboard = TRUE;
        break;
      }
    }
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  if ( !found_hexboard ) return FALSE;  // never found the hexboard

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
double sim_get_time_ratio( )
{
  DEBUG_INIT( "SIMULATE" , "sim_get_time_ratio" );
  DEBUG_OUT( "entering", 1 );
```

```
  return time_ratio;
}
////////////////////////////////////////////////////////////////////////
int sim_get_time_slice( )
{
  DEBUG_INIT( "SIMULATE" , "sim_get_time_slice" );
  DEBUG_OUT( "entering", 1 );

  return time_slice;
}
////////////////////////////////////////////////////////////////////////
int sim_model_close( )
{
  DEBUG_INIT( "SIMULATE" , "sim_model_close" );
  DEBUG_OUT( "entering", 1 );

  // closing the current model just involves removing the name from
  //  "current_model_name"
  if ( current_model_name ) {
    free( current_model_name );
    current_model_name = NULL;
  }
  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_model_list_of_existing( char ***models, int& size)
{
  DEBUG_INIT( "SIMULATE" , "sim_model_list_of_existing" );
  DEBUG_OUT( "entering", 1 );
  static char **existing_models = NULL;

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    existing_models = (char**)XtRealloc( (char*)existing_models,
      model::extent->cardinality( ) * sizeof(char*) );
    size = 0;
    foreach ( model *m, *model::extent ) {
      existing_models[size++] = strdup( m->name );
    }
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  *models = existing_models;
  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_model_new( char* model_name, double& elapsed_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_model_new" );
  DEBUG_OUT( "entering", 1 );
  int already_exists = FALSE;

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    // ensure that the new model name is not already in use by another model
    foreach ( model *m, *model::extent ) {
      if ( !strcmp( model_name, m->name ) )
        already_exists = TRUE;
```

```
    }
  }
  if ( already_exists ) return FALSE;
  do_transaction () {
    os_configuration *model_config = new(db) os_configuration( );
    new (db, model_config) model( model_name, model_config );
    // save the name so the model may be attached to when needed
    current_model_name = strdup( model_name );
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  r   rn TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_model_open( char* model_name, double& elapsed_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_model_open" );
  DEBUG_OUT( "entering", 1 );

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    // make sure that the model exists in the database
    model *current_model = ( *model::extent )[%strcmp(name,model_name)==0%];
    // save the name so the model may be attached to when needed
    current_model_name = strdup( model_name );
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_model_save_as( char* model_name, double& elapsed_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_model_save_as" );
  DEBUG_OUT( "entering", 1 );

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
  do_transaction () {
    // copy the model, but don't change the current model
    os_configuration *model_config = new(db) os_configuration( );
    new (db, model_config) model( get_current_model( ), model_name, model_config );
  }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  return TRUE;
}
```

```
////////////////////////////////////////////////////////////////////////
int sim_report( double& elapsed_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_report" );
  DEBUG_OUT( "entering", 1 );
  FILE *rfile;

  if ( ( rfile = fopen( "SIM_REPORT" , "wt" ) ) == NULL ) {
    return FALSE;
  }

  stopwatch timer;
  timer.start( );

  TIX_HANDLE(all_exceptions)
    do_transaction () {
      model *current_model = get_current_model( );

      // report all the logitems
      fprintf( rfile, "-------------------------------------------------\n" );
      fprintf( rfile, " AFIT SimBench REPORT (SIM TIME: %d sec)\n",
        current_model->sim_time );
      fprintf( rfile, "-----------+-----------+-----------+-----------\n" );
      fprintf( rfile, "%10s | %10s | %10s | %10s\n",
        "AIRCRAFT", "TRUCK", "LOCATION", "SIM TIME" );
      fprintf( rfile, "-----------+-----------+-----------+-----------\n" );
      for ( logitem *l = current_model->first_logitem ; l ; l = l->next ) {
        l->output( rfile );
      }
      fprintf( rfile, "-----------+-----------+-----------+-----------\n" );
    }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  elapsed_time = timer.stop( );

  fclose( rfile );

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
int sim_run_until_sim_goal_time( int goal_sim_time )
{
  DEBUG_INIT( "SIMULATE" , "sim_run_until_sim_goal_time" );
  DEBUG_OUT( "entering", 1 );

  TIX_HANDLE(all_exceptions)
    do_transaction () {
      model *current_model = get_current_model( );
      current_model->run_until_sim_goal_time( goal_sim_time );
    }
  TIX_EXCEPTION
    return FALSE;
  TIX_END_HANDLE

  return TRUE;
}
////////////////////////////////////////////////////////////////////////
void sim_set_time_ratio( double new_time_ratio )
{
  DEBUG_INIT( "SIMULATE" , "sim_set_time_ratio" );
  DEBUG_OUT( "entering", 1 );
```

```
  time_ratio = new_time_ratio;
}
////////////////////////////////////////////////////////////////////
void sim_set_time_slice( int new_time_slice )
{
  DEBUG_INIT( "SIMULATE" , "sim_set_time_slice" );
  DEBUG_OUT( "entering", 1 );

  time_slice = new_time_slice;
}
////////////////////////////////////////////////////////////////////
```

3.1.23 truck.hh

The source code for the file *truck.hh* is listed below.

```
#ifndef ___TRUCK_HH
#define ___TRUCK_HH
/*

#######
   #     #####   #    #   ####   #    #
   #     #    #  #    #  #    #  #   #
   #     #    #  #    #  #       ####
   #     #####   #    #  #       #  #
   #     #   #   #    #  #    #  #   #
   #     #    #   ####    ####   #    #

 truck.hh

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993
*/
#include"player.hh"

class truck : public player {

  char name[10];
  char type[12];
  char payload[12];
  char location[10];
  char misc_sim_data[51];

public:

  truck( char *name, char *type, char *payload, char *location,
    char *misc_sim_data );
  environment *clone( os_database *db, os_configuration *model_config );
  void execute( char *message, model* sim_model );
  char *query( char *data_item );
};
#endif ___TRUCK_HH
```

3.1.24 truck.cc

The source code for the file *truck.cc* is listed below.

```
/*

#######
   #     #####   #    #   ####   #    #
   #     #    #  #    #  #    #  #   #
```

```
   #     #    #  #    #  #       ####
   #     #####   #    #  #       #  #
   #     #   #   #    #  #    #  #  #
   #     #    #   ####    ####   #    #

 truck.cc

 Air Force Institute of Technology
 Timothy J. Halloran
 20 Aug 1993

 Revisions:
 30 Sep 1993 -TJH- Removed all class inline functions due to compiler
                   limitations (CC compiler is not a "full" C++).
*/
#include<ostore/ostore.hh>
#include<ostore/coll.hh>
#include<ostore/relat.hh>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<debug.hh>
#include"model.hh"
#include"truck.hh"
#include"dice.hh"
#include"expon.hh"
#include"location.hh"

#define STREAM 1
////////////////////////////////////////////////////////////////////////
truck::truck( char *name, char *type, char *payload, char *location,
  char *misc_sim_data )
{
  DEBUG_INIT( "TRUCK" , "truck::truck" );
  DEBUG_OUT( "entering", 1 );

  strncpy( truck::name, name, 9 );
  truck::name[9] = '\0';
  strncpy( truck::type, type, 11 );
  truck::type[11] = '\0';
  strncpy( truck::payload, payload, 11 );
  truck::payload[11] = '\0';
  strncpy( truck::location, location, 9 );
  truck::location[9] = '\0';
  strncpy( truck::misc_sim_data, misc_sim_data, 50 );
  truck::misc_sim_data[50] = '\0';

  DEBUG_OUT( sprintf( BUG, "%s %s %s %s", name, type, payload, location ), 2 );
}
////////////////////////////////////////////////////////////////////////
environment *truck::clone( os_database *db, os_configuration *model_config )
{
  DEBUG_INIT( "TRUCK" , "truck::clone" );
  DEBUG_OUT( "entering", 1 );

  environment *c = new(db, model_config)
    truck( name, type, payload, location, misc_sim_data );
  return c;
}
////////////////////////////////////////////////////////////////////////
void truck::execute( char *message, model* sim_model )
{
  DEBUG_INIT( "TRUCK" , "truck::execute" );
  DEBUG_OUT( "entering", 1 );
```

```
  DEBUG_OUT( sprintf( BUG, "message is \"%s\"", message ), 2 );
  if ( !strcmp( message, "MOVE" ) ) {
    // execute the "MOVE" event
    int hexboard_width, hexboard_height;

    // determine the size of the hexmap
    for ( environment *e = sim_model->first_member ; e ; e = e->next ) {
      if ( !strcmp( e->query( "NAME" ), "HEXBOARD" ) ) {
        hexboard_width = atoi( e->query( "WIDTH" ) );
        hexboard_height = atoi( e->query( "HEIGHT" ) );
        break;
      }
    }

    // move the truck
    random_move( location, hexboard_width, hexboard_height );

    // schedule the next move for this truck (exponential with mean 600.0)
    sim_model->schedule( this, name,
      sim_model->sim_time + (int)expon( 600.0 , STREAM ),
      "MOVE" );
  }
  else {
    fprintf( stderr, "WARNING[truck-%s] unknown message\n", name );
  }
}
////////////////////////////////////////////////////////////////////////
char *truck::query( char *data_item )
{
  DEBUG_INIT( "TRUCK" , "truck::query" );
  DEBUG_OUT( "entering", 1 );

  if ( !strcmp( data_item, "CLASS" ) ) {
    return "TRUCK";
  }
  else if ( !strcmp( data_item, "NAME" ) ) {
    return name;
  }
  else if ( !strcmp( data_item, "TYPE" ) ) {
    return type;
  }
  else if ( !strcmp( data_item, "PAYLOAD" ) ) {
    return payload;
  }
  else if ( !strcmp( data_item, "LOCATION" ) ) {
    return location;
  }
  return "ERROR";
}
////////////////////////////////////////////////////////////////////////
```

References

[1] Cattell, R.G.G. *Object Data Management Object-Oriented and Extended Relational Database Systems*. Reading, MA: Addison-Wesley Publishing Company, 1991.

[2] Halloran, Timothy J. *Performance Measurement of Three Commercial Object-Oriented Database Management Systems*. MS thesis, AFIT/GCS/ENG/93D-12, Department of Engineering, Air Force Institute of Technology, December 1993.

[3] Itasca Systems. *C++ API User Manual for Release 2.2*. Minneapolis, MN: Itasca Systems, Inc., 1993.

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5 November 1993	Technical Report

4. TITLE AND SUBTITLE
Source Code for the OO1 Benchmark and the AFIT Simulation Benchmark

5. FUNDING NUMBERS

6. AUTHOR(S)
Timothy J. Halloran, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/EN-TR-93-09

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This paper contains the source code for implementations of the OO1 benchmark and the AFIT simulation benchmark. OO1 benchmark implementations are listed for the Itasca, Matisse, and ObjectStore object-oriented database management systems (DBMS). An AFIT simulation benchmark implementation is listed for the ObjectStore object-oriented DBMS.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Database Benchmarks, Object-Oriented Databases, Simulation	144
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL